Université Gustave Eiffel

Multi-Agent Systems

EL FENNI Hassan FREDJ Najeh

M2 SIA

Supervised by :

Mr. Mahdi Zargayouna

2020/2021

Contents

INTRODUCTION

Distributed computing is of great significance in current and future reserach and applications of computer science. As a consequence, in artificial intelligence we can observe rapid growth in the subfield of distributed artificial intelligence (DAI). In particular research on multiagent systems and their potential applications currently attract a lot of interest.

COVID-19 has been spreading around the world. Many countries are on lockdown to mitigate the spread of this deadly virus. Everyone around the world has adopted the practice of social distancing (or isolation). We will be going through a popular agent-based modelling Java framework — Repast Simphony. We will attempt to build a **basic** simulation to monitor the spread of coronavirus in various settings.

1 Project framework

1.1 Problematic

In the context of applying our knowledge about artificial intelligence and use of the multi-agent systems we have chosen the project 2: Covid 19 simulation.

1.2 Objective

The main objective of this project is to represent the propagation of a Covid-19 like pandemic and to simulate various movement scenarios of people (random movements, random movements with central attractors (malls, schools, workplaces, etc) and various strategies for pandemic limitation (lockdown, curfew, isolation of infectious persons, etc).

1.3 Tools and technologies

1.3.1 Eclipse

Eclipse is an integrated development environment (IDE) used in computer program- ming.It contains a base workspace and an extensible plugin system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plugins.

1.3.2 Java

Java is an object oriented programming language. A special feature of Java is that software written in this language is compiled to an intermediate binary representation that can be executed in a Java Virtual Machine (JVM) without regard to the operating system.

1.3.3 Repast Simphony

The Recursive Porous Agent Simulation Toolkit (Repast) is a widely used free and open-source, cross-platform, agent-based modeling and simulation toolkit. Repast has multiple implementations in several languages (North, Collier & Vos 2006) and built-in adaptive features, such as genetic algorithms and regression.

Repast was originally developed by David Sallach, Nick Collier, Tom Howe, Michael North and others at the University of Chicago.

2 Realisation

2.1 Design choices

2.1.1 Our agents:

1. Healthy Agents
2. Infected Agents
3. Recovered Agents
4. Dead Agents
5. Hospitals
6. Mall
7. School
8. Workplace

2.1.2 Interaction modes:

The simulation starts with loading our initialization parameters (limitation strategy,movement scenario, number of Healthy and Infected agents…), the context builder is responsible for creating the corresponding agents as well as the simulation environment (grid & continuous space). The scheduler is responsible for synchronization of agent execution at each tick of simulated time. We have used Repast Simphony annotations to manage all of our scheduled methods. Once launched, each agent executes a method step. The behavior of each type of agent is described in the control system section. Finally, when the simulation duration is reached, the results are collected and the simulation ends.

2.1.3 Representation of the environment:

We represent the simulation environment using:

-Grid

-Continuous Space

2.2 The structuring of the program

2.1.3 Control system:

In order to be able to implement limitation strategies such as curfew, we have opted for **a tick of one hour**.

To initialize and control the simulation, we used the following parameters:

-GridHeight: Allows to specify the height of our grid

-GridWidth: Allows to specify the width of our grid

-Healthy Count: Initialize the number of Healthy people in the simulation

-Infected Count: Initialize the number of Infected people in the simulation

-Limitation Strategy Selection: Allows us to select which pandemic limitation strategy to adopt for the simulation run, we can choose a strategy from the 4 options available:

- none: No strategy to adopt
- curfew: Agents are able to move around during daytime hours. However, from 20pm to 6am they are not allowed to leave their respective houses.
- lockdown: No one is allowed to leave their house.
- isolateInfectious: Infected agents are not allowed to leave their houses.

-Movement Scenario Selection:

- moveRandomly: Agents move randomly within the grid
- centalAttractors: Agents will hover around the closest central attractor to them (school, workplace or mall) depending on their situation.

  If the agent is a student, he will move randomly within the vicinity of his school.

  If the agent works, he will move randomly within the vicinity of his workplace.

- Number of hospitals: Specifies the number of hospitals in the simulation.
- Number of rooms in each hospital: Specifies the capacity of each hospital.
- Number of Malls: Specifies the number of Malls in the simulation.
- Number of Schools: Specifies the number of Schools in the simulation.
- Number of Workplaces: Specifies the number of workplaces in the simulation.
- Probability to wear mask: Probability that any agent will wear a mask.
- Probability to die: Fixed probability of death for all Infected agents, we realize this solution as well as the next 2 parameters aren't optimal and we are aiming to improve it to better represent reality (impact of age & chronic diseases).
- Probability to recover in hospital: Fixed probability of recovery after going to hospital for all Infected agents.
- Probability to recover naturally: Fixed probability of naturally recovering for all agents.
- Transmission rate: Probability that an Infectious agent will transfer the virus when he comes in contact with a Healthy agent. this probability will change within the code if either or both agents wear a mask.

2.1.4 Agents:

Our simulation consists of 2 types of agents:

-Agents that represent the different states a person could be in: Healthy, Infectious (with or without symptoms), Dead or Recovered (immune to the virus).

-Agents that represent the different central-attractors (Mall, Workplace, School) as well as the Hospital agent.

Agents that represent a human:

All the agents except Dead share a step() function scheduled at each timestep, in which we will implement the different movement scenarios according to the limitation-strategy chosen.

- **Infected agent:** Our infected agent has 4 scheduled methods, which are:

  step() scheduled at each time step with a priority of **2**, the step method allows us to implement the various movement strategies as well as the limitation-stragies, it also deals with infecting other agents.

  go_to_hospital() scheduled at each time step, with a priority of **1**.

  This method takes an infected agent to the hospital, it could lead to a final state which is the recovery of the agent, therefore it must have the highest priority.

  recover_naturally() scheduled at each time step, with a priority of **1**.

  This method could lead to a final state which is the recovery of the agent, therefore it must have the highest priority.

  die() scheduled at each time step, with a priority of **1**.

  This method could lead to a final state which is the death of the agent, therefore it must have the highest priority.

- **Healthy agent:** Healthy agent has 1 scheduled method:

  Step() scheduled at each time step with a priority of **1**, the step method allows us to implement the various movement strategies as well as the limitation-strategies.

- **Recovered agent:** Recovered agent has 1 scheduled method:

  Step() scheduled at each time step with a priority of **1**, the step method allows us to implement the various movement strategies as well as the limitation-strategies.

- **Dead:** Dead agent has no scheduled methods.

2.3 The difficulties encountered

We have encountered several difficulties during the realization of this project which are:

-The scheduling of our different methods, scheduling methods in a consistent way that doesn't lead to unwanted behaviors.
-Parameter tuning in order to accurately represent reality, agents are assigned an age attribute based on real age distribution probabilities for example.
-In order to further improve our work, we aim to implement realistic probabilities of death and recovery as stated before, these probabilities would need to be function of the agent's age and the fact that he has a chronic disease or not for example. We also aim to add a policy Compliance factor such that not all agents comply to the limitation strategies.
Moreover, we could implement various other methods such as goHome, sleep, wakeUp as well as custom realistic maps to represent a real city and be able to compare our simulation results with real life data.
We have been working on these and other possible improvements, however time was of the essence here.

2.4 User guide

Step1: Repast Simphony comes with its own Java Environment (Eclipse)

Launch the eclipse IDE, Import the project, Switch perspective to java perspective then finally the project (Covid19-SimulationModel).

Step2: Setup the parameters:

We can fix several parameters as shown below:

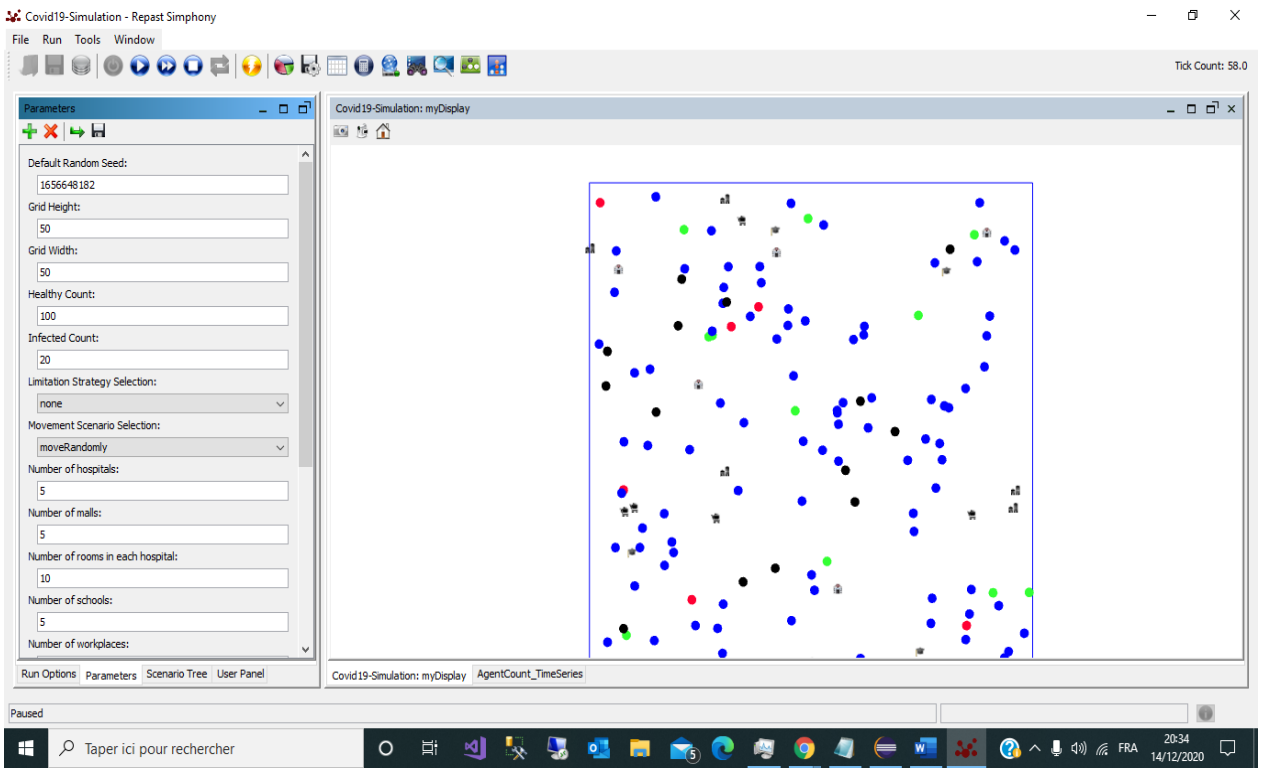

Figure 1: SimulationGUI

We have two movement scenarios: moveRandomly & centralAttractors

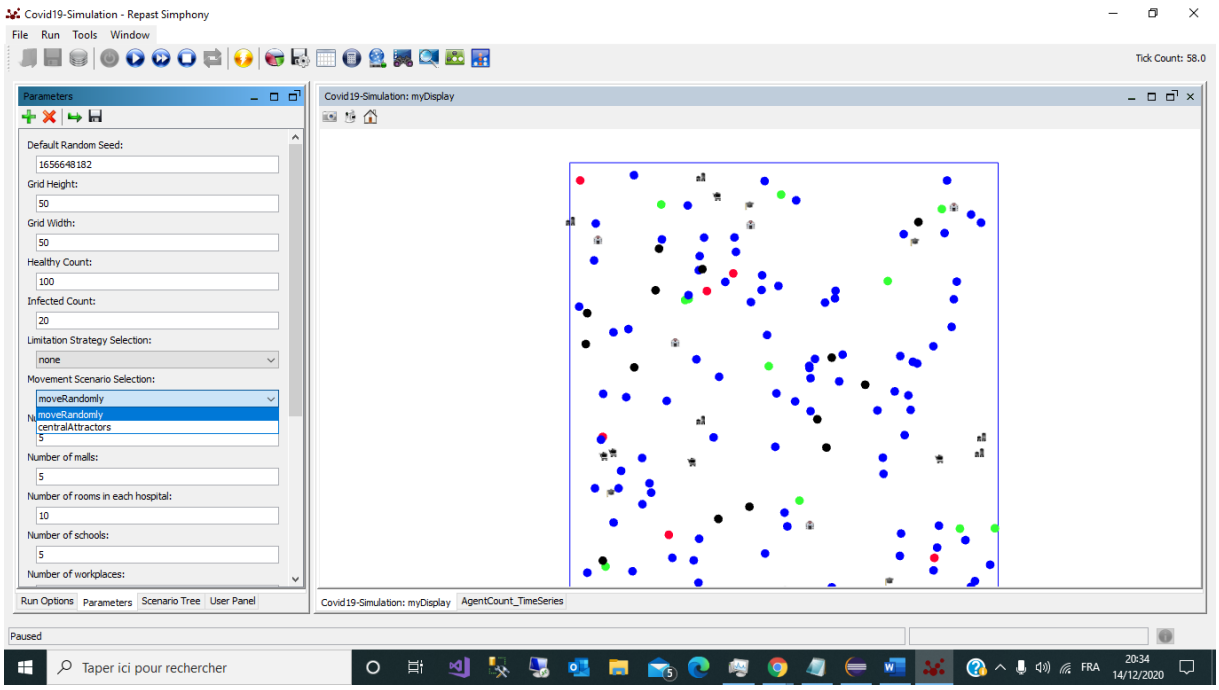

Figure 2: Movement Options

We have 4 limitation strategies: none, curfew, lockdown & isolateInfectious

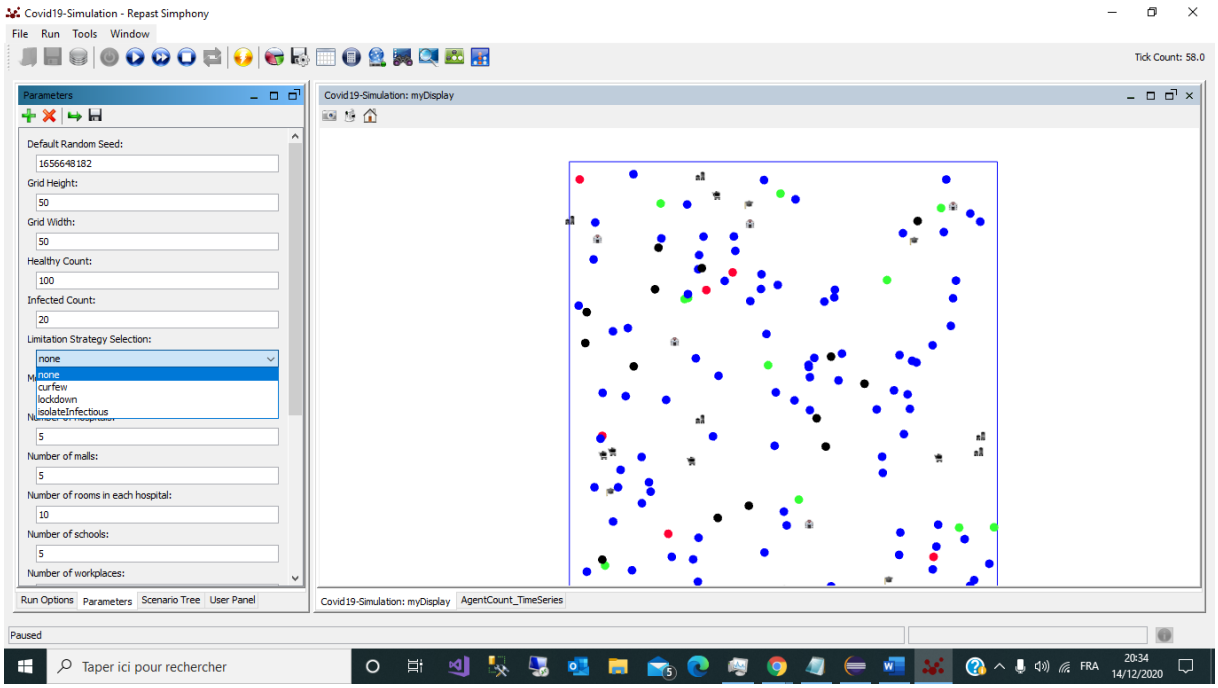

Figure 3: Strategies

Step 3: Run the simulation.

Notes:
Blue: Healthy
Red: Infected
Green: Recovered
Black: Dead

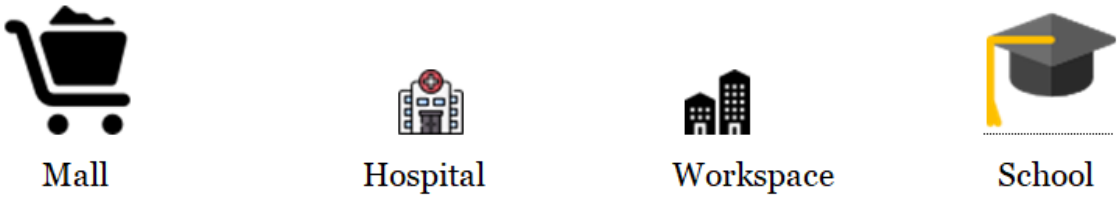

Figure 4: Icons

Example :

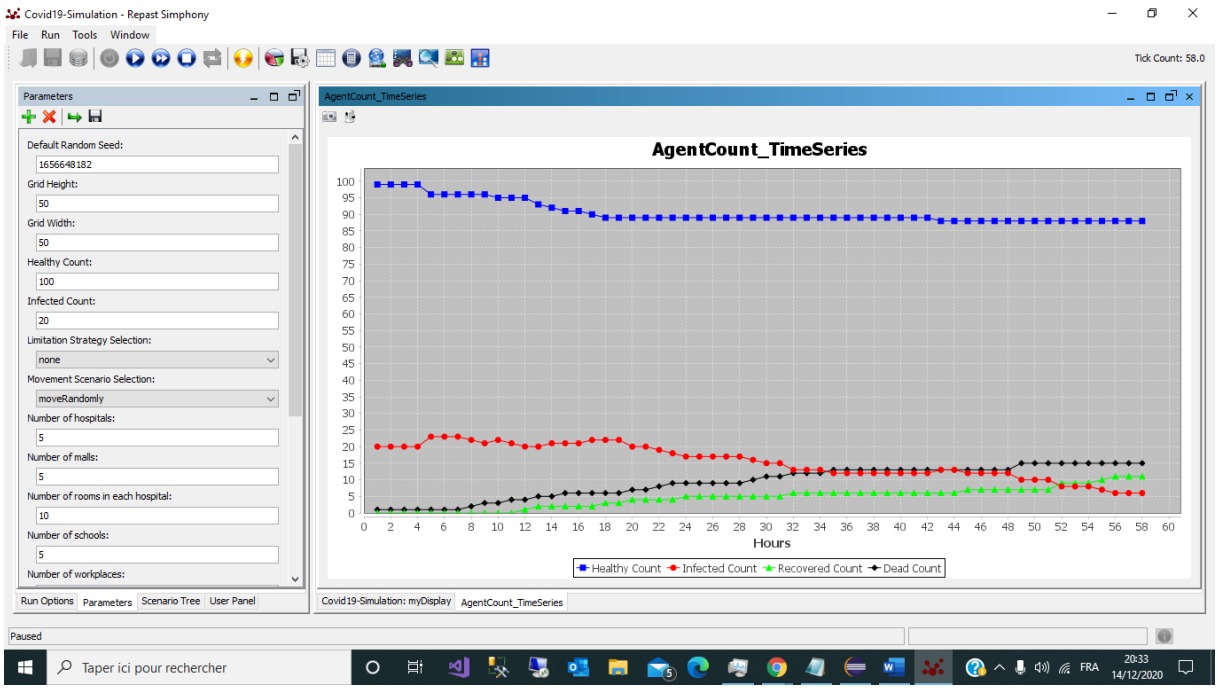

Figure 5: Example output

2.5 Results:

We ran our simulation using the following parameters while trying out all different pandemic limitation strategies:

gridHeight = 200 gridWidth = 200

HealthyCount = 10000 , InfectedCount = 200

Movement scenario = moveRandomly

Number of hospitals & malls = 50

Number of rooms per hospital = 20

Number of schools & workplaces = 20

Probability to go to hospital = 0.5

Probability to recover after going to hospital = 0.5

Probability to recover naturally = 0.01

Probability to wear mask = 0.8

Transmission rate = 0.8

strategy = none

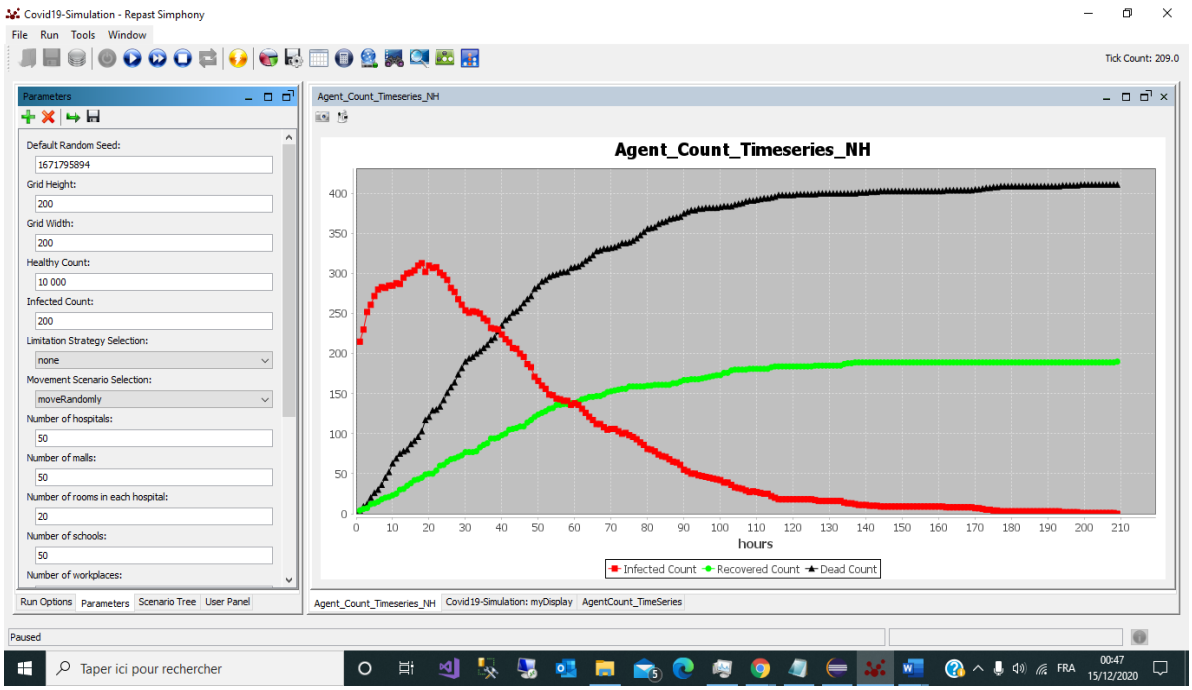

Figure 6: AgentCountNH_none

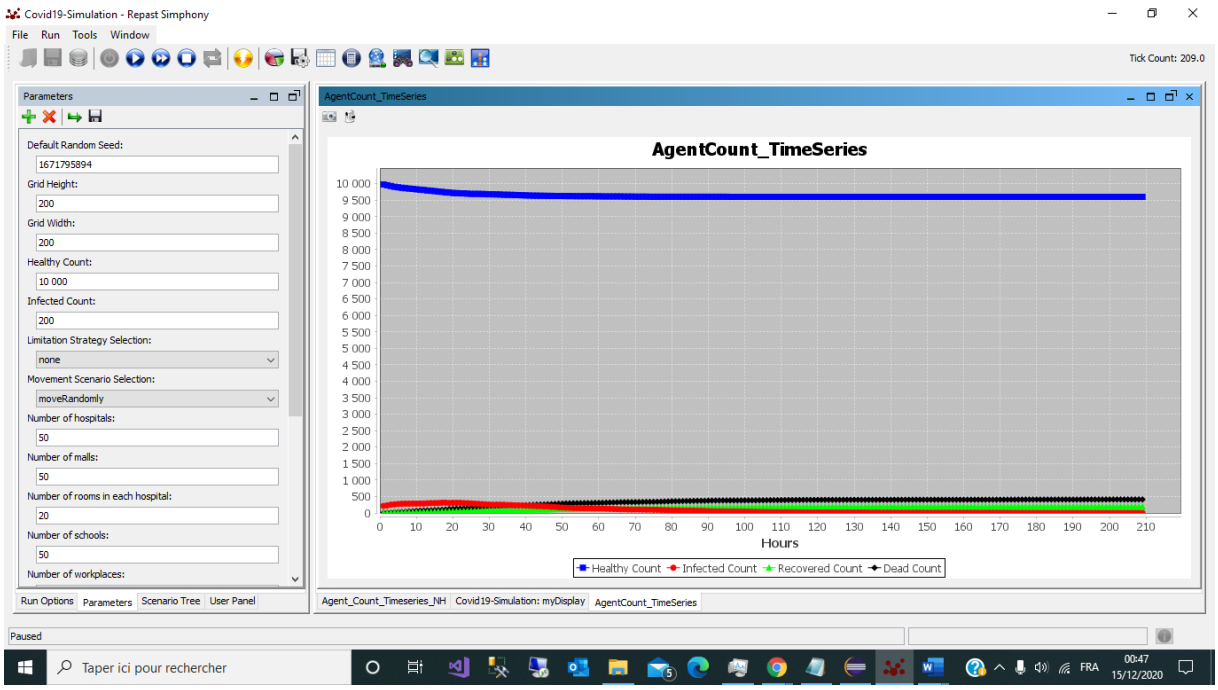

Figure 7: AgentCount_none

strategy = curfew

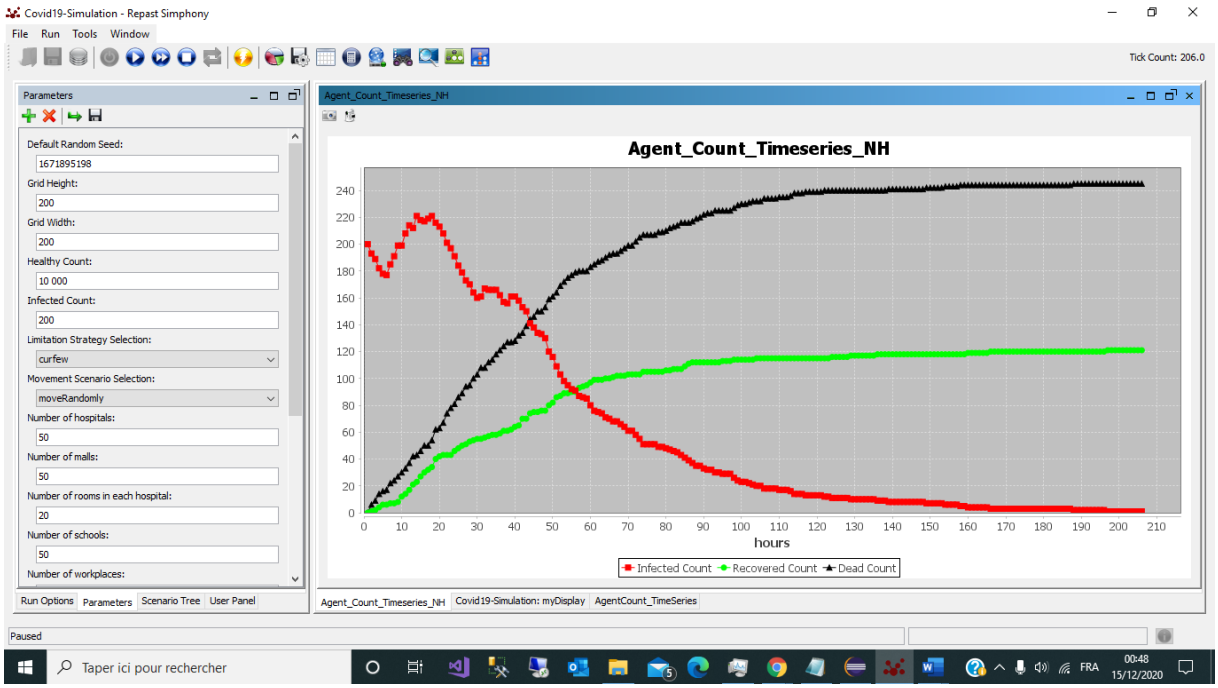

Figure 8: AgentCountNH_curfew

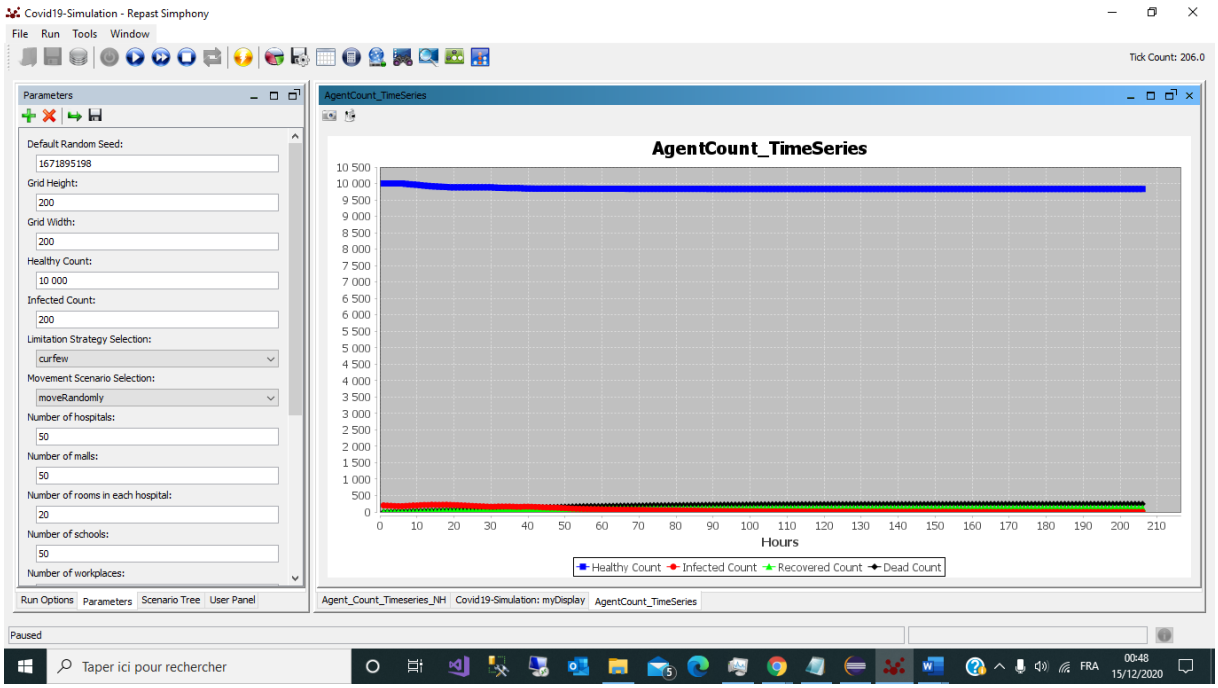

Figure 9: AgentCount_curfew

strategy = lockdown

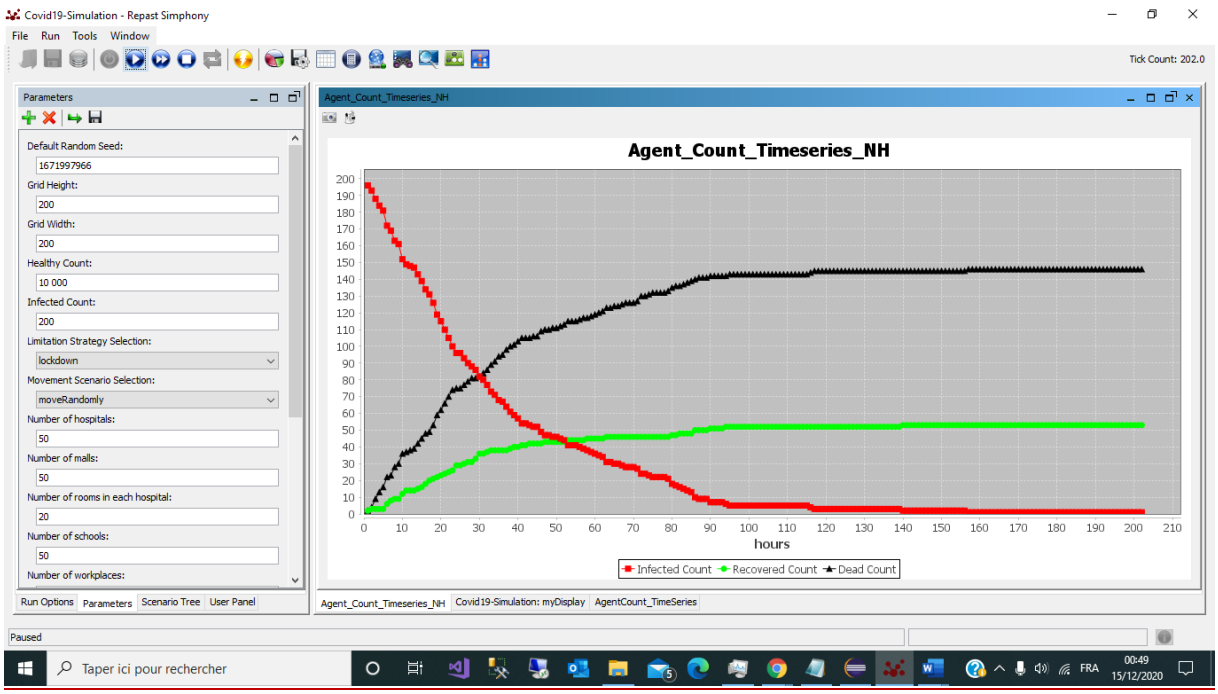

Figure 10: AgentCountNH_lockdown

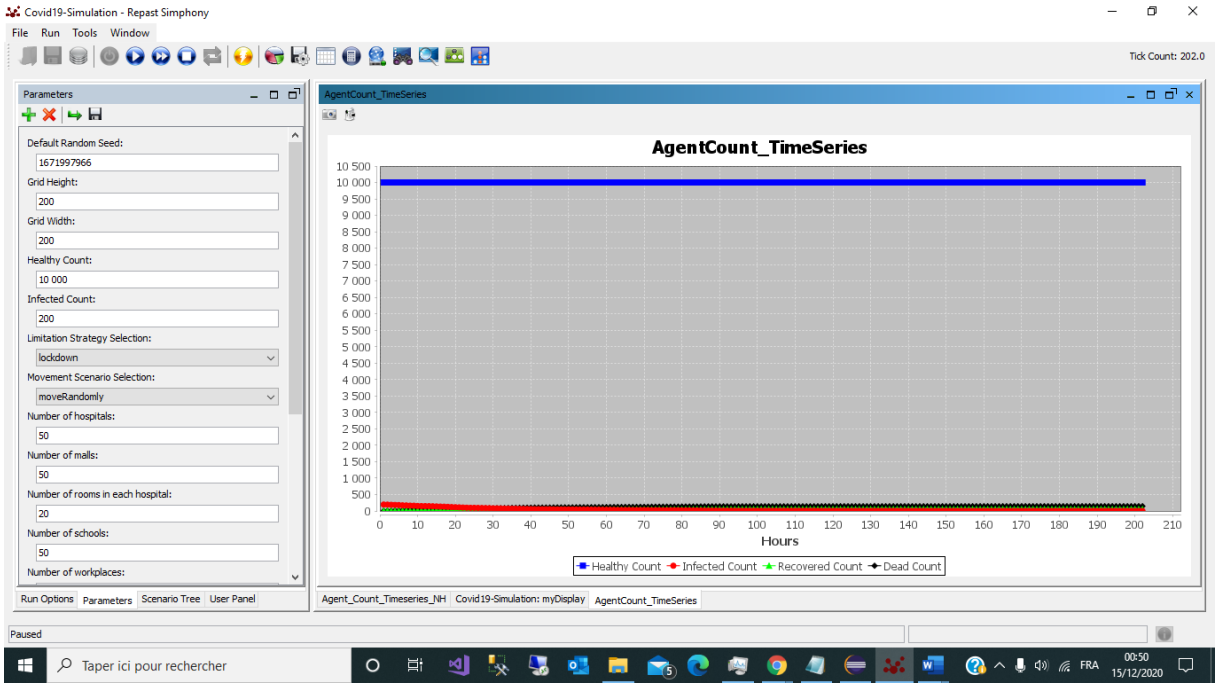

Figure 11: AgentCount_lockdown

strategy = isolateInfectious

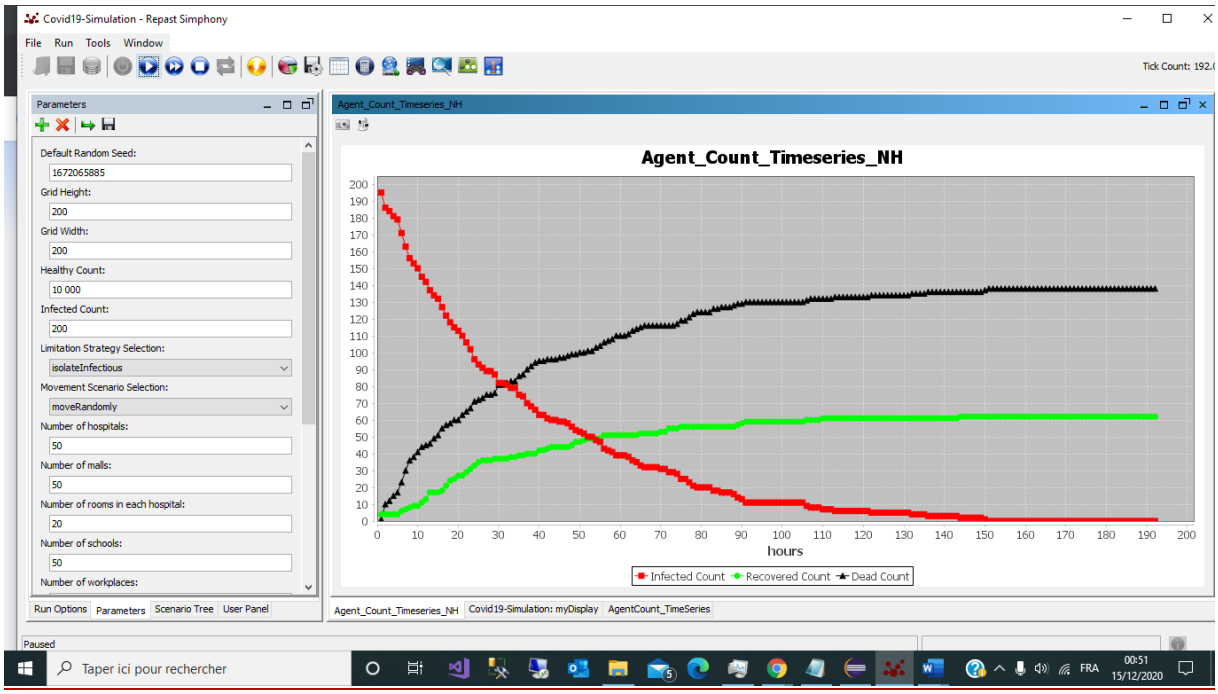

Figure 12: AgentCountNH_isolation

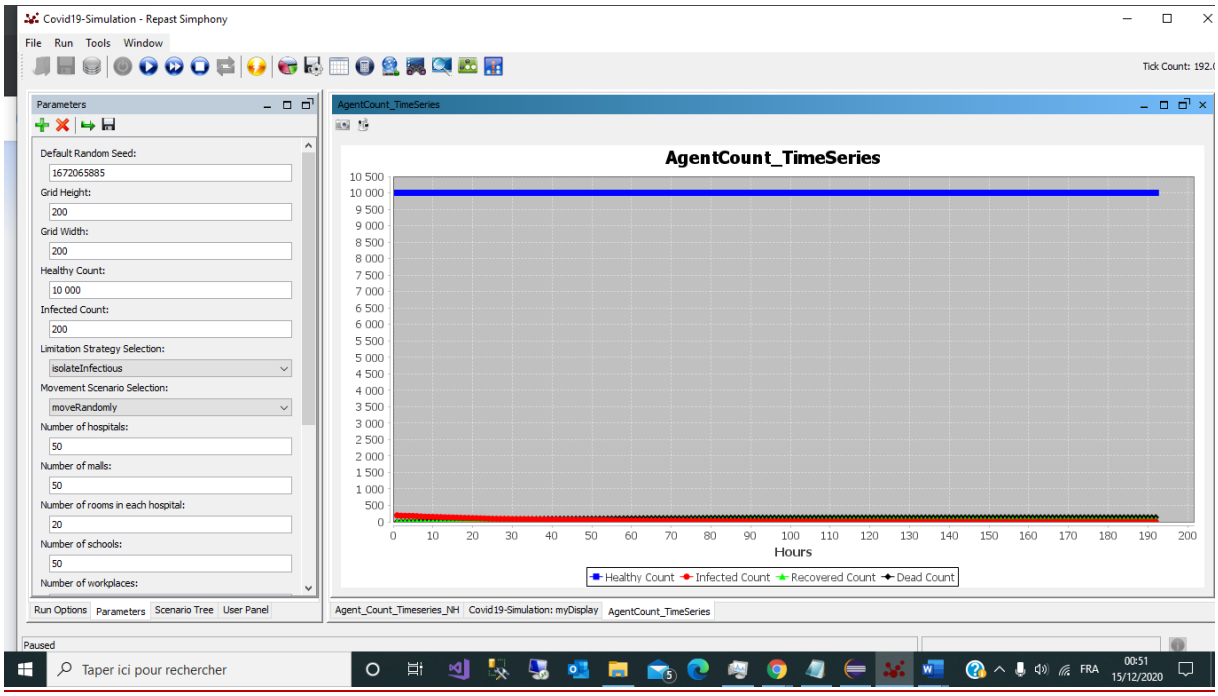

Figure 13: AgentCount_isolation

The results show that the best 2 strategies are by far the total lockdown strategy as well as the strategy of isolating infectious people. Surprisingly, isolating infectious people actually results in slightly less deaths than a total lockdown.

Conclusion

We have built a simulation that allows us to simulate the propagation of a covid-19 like pandemic, under different movement and prevention strategies. The aim of this project is to offer a simple tool for policy makers to compare different strategies in terms of when to implement the strictest containment measures and how different treatments can increase or suppress infected cases.

Results have shown that, surprisingly the most effective pandemic limitation strategy is a complete lockdown of the infectious people as it greatly reduces the spread of the virus and provides time for hospitals to treat more patients. However, we realize that such a measure is nearly impossible in real life as it is very hard to identify all covid-19 cases at a given time.

In the near to mid future, we aim to improve this project in order to obtain more realistic results and compare it with real life data. as discussed in the difficulties encountered section.

This project has been very enriching to us as it has taught us a lot about the Repast Simphony simulator, a tool that could prove to be useful in the future.