

Using AI to Develop Emergent Ways to Prevent Disease Spread

James Absolom

Robotics - Falmouth University

Aim of artefact

After recent human events it's become increasingly important to come up with new ideas to prevent disease spread through communities. The aim of my AI is to show unique emergent ways of meeting important needs while also avoiding the spread of the disease to either form a better defense against disease spread or confirm our current tactics are suitable.

Introduction

To fulfill the aim of the artefact I have made multiple components which will make a full genetic system, these components are:

- An agent brain to poll the environment and then tell the agent what action to take next.
- A genetic learning algorithm to train the agent brains to fulfill the task.
- An environment for the agents to train in with food and water areas as well as walls to restrain the agents.

The project has been coded in Godot as it is a really lightweight game engine which uses GDScript [1] which is a python like programming language which I am very confident with. This project uses specifically Godot 3.2 which is a few versions out of date now but works well for what I need, and after some testing it still works on Godot 4.0 which was released at the start of march 2023 [2]. If you wish to just run the AI and see it working run go to the releases area on the Github (below) and download the version for your OS.



Figure 1:QR Code link to repo

Link

You can also just follow the link to find the repo [github.falmouth.ac.uk/JA244121/Comp250-JA244121](https://github.com/falmouth.ac.uk/JA244121/Comp250-JA244121).

Agents

Each individual agent has several parts which control the actions and movements it takes, these are:

- A kinematic body which allows for movements and collisions inside the environment, this currently allows the agents to collide with the exterior walls of the environment but makes sure the agents don't collide with each other as this caused quite a few issues in earlier versions.
- A collision polygon which works along side the kinematic body to calculate collisions.
- An action controller script which acts like a state machine swapping between different states depending on what variables it is fed, this script also runs all the movement vector calculations, having this split into a different script also slightly increases the systems performance as it can run the calculations in parallel as apposed to queuing it after the polling actions.
- A brain script which polls the environment around the agent and works out the next action based on trained values and the results from polling the environment.
- 2 simple circle sprites, one white and one blue to easily visualise the spread of the disease for the user.

Genetic Training

To train the agents I'm running genetic algorithm which keeps and mutates well performing agents while getting rid of agents which score poorly. The score of the agents is adjusted by multiple factors:

- +1 score for every second the agent is alive and not infected
- -500 points for getting infected
- -1000 points for dying via lack of food or water

These values can of course be adjusted to fit the users priority. The agents are then sorted by their scores with the best scoring agents being highest on the list. The top 1/3 of this list is then taken and copied across to the list for the next epoch. This top 1/3 is then mutated by a random amount capped by a max mutation variable and added to the next epoch list. The copy and then mutate takes up the top 2/3 of the next epoch so the remaining 1/3 is set by random values meaning we avoid over fitting one solution and look for other ways of solving the problem.

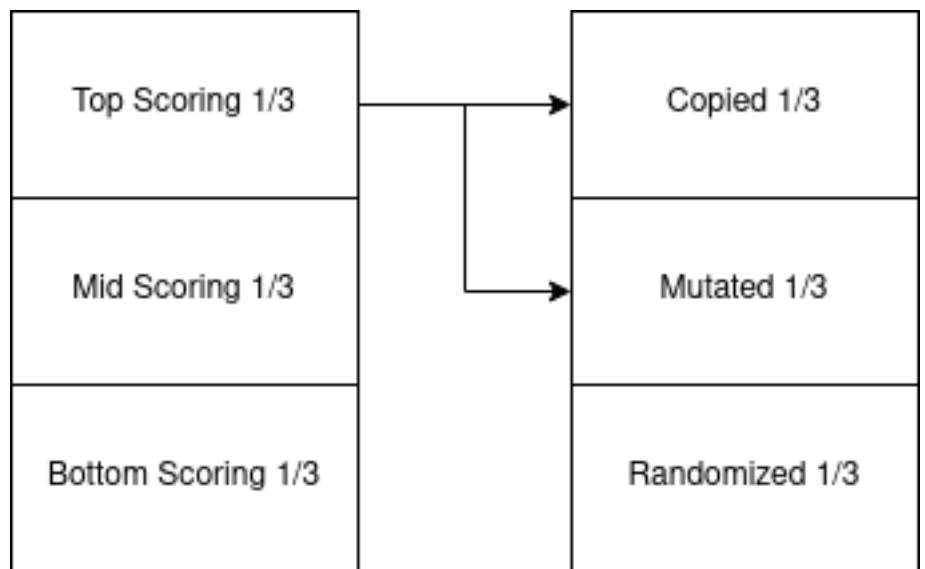


Figure 2:A diagram to show the genetic learning

Mathematical Equations

Each action has a priority, these priorities are the compared to work out which action the agent should do next; these priorities are calculated by custom written equations which look something like this:

$$\frac{(1 - (\frac{MaxValue_1}{CurrentValue_1})) * multiplier_1 + (1 - (\frac{MaxValue_2}{CurrentValue_2})) * multiplier_2}{2} \quad (1)$$

Note: this is generalised, the equation changes for some parts in the code

The max values are set as global variables and define the maximum a value can be, for example the total amount of agents in the system or the maximum amount of food an agent can have. These values are then divided by the current value which the agent gets when it polls the environment. The equation then takes this value away from one to standardise all the values to between 0 and 1 and make it so higher the values correctly correspond to how they should react. The equation then gets the average by adding the multiplied values together and dividing them by how many values there are. This means multiple values can lead to one action as it will just be an averaged out. Here is a diagram of the whole process to hopefully make it easier to understand:

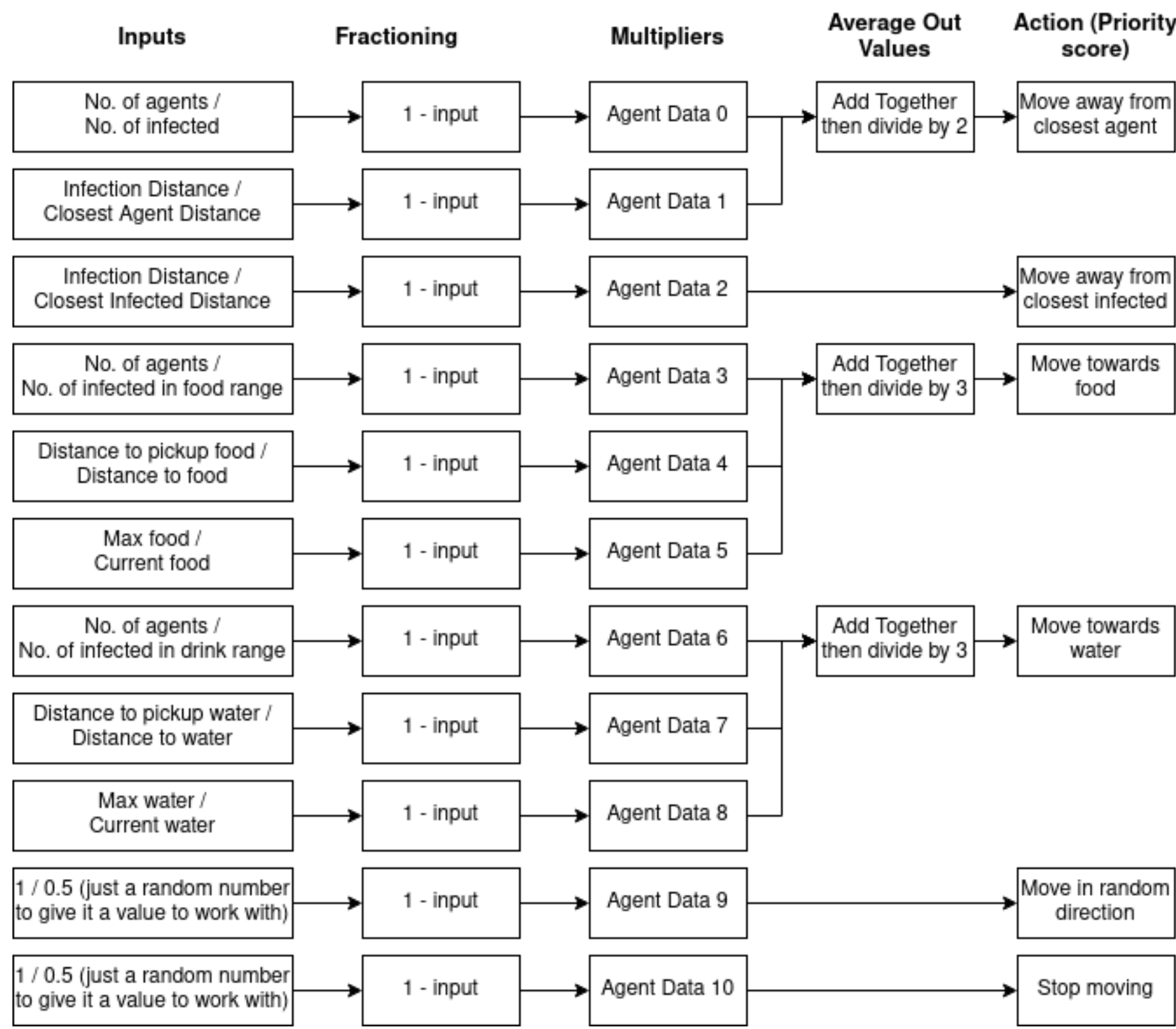


Figure 3:A diagram to show the equations

Future changes

If I were to develop this project further I would move away from its current decision making methodology and return to a more traditional neural net. I also would like to move from a traditional genetic algorithm to a NEAT algorithm which allows the AI to change the composition of the neural net. I already know of a library for Godot by Paul Straberger which adds NEAT implementation which I'd like to base my work off of [3].

Conclusion

The developed system works well enough however it still has flaws for example the food and water sources just end up becoming major spreaders of the disease. This is mostly because of the poll rate of the agents causing them to sit in the food and water sources while waiting for a poll. There are two main solutions I can see to fixing this issue; I could make it so agents poll more frequently and just ignore the lag that that would cause in the system, or I could add more food and water sources to make it so they have choice in which source they go to. Despite both these conclusions the genetic learning system did come up with some unique emergent behaviours. The solution they generally came up with was to run in for food and water and then instantly run into the exterior wall to ensure they are as far away from every agent as possible. Though it is difficult to decipher entirely what this solutions real life parallels are I believe it may suggest stockpiling supplies at the start of an infection may lead to higher chance at avoiding the disease. We saw this ideal at the beginning of the Covid19 pandemic where stores saw record sales of goods "exceeding even peak Christmas trade" [4]. This draws an interesting parallel to real life and poses the question "what if we added supply shortages into the system?".

References

- [1] "Gdscript reference." https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html, accessed 17-03-2023.
- [2] G. Engine, "Godot 4.0 sets sail: All aboard for new horizons." <https://godotengine.org/article/godot-4-0-sets-sail/>, accessed 17-03-2023.
- [3] pastra98, "Pastra98/neat_for_godot : An implementation of kennetho.stanley's nea algorithm for the godot game engine." https://github.com/pastra98/NEAT_for_Godot, accessed 19-03-2023.
- [4] X. Friedländer, "Covid-19: Shopper buying behavior and out-of-stock products." <https://www.retailinsight.io/blog/covid-19-shopper-buying-behaviour-and-out-of-stock-products>, accessed 20-03-2023.

Contact Information

- Email: ja244121@falmouth.ac.uk
- Github: [github.falmouth.ac.uk/JA244121](https://github.com/falmouth.ac.uk/JA244121)

