Utrecht Summer School

Introduction to Complex Systems

Project Day 4

Evolution in an IBM

1 Population model and evolution

In this last project, we will explore the process of evolution in a simple model. For this, we will use an individual based model (abbreviated often to IBM), which is also sometimes called an agent based model (abbreviated often to ABM). The IBM in this project can be thought of as a population model.

The IBM in this project models a number of different individuals (or 'agents') that live in a spatial domain that get a limited amount of food. If an individual has no access to food, it has a chance to die. If it has access to food, it won't die and there is a chance it will create offspring. Additionally, the individuals can move through space. The probabilities of dying and reproduction, as well as the motility, typically vary between individuals depending on their traits (i.e., the parameters of the individual in the IBM).

To model this, we consider a two-dimensional domain with periodic boundary conditions (e.g., leaving the domain on the right brings you back in the domain on the left). On this domain, we model a certain amount of individuals that each have their own motility m, death rate d and growth rate r. At every time step we introduce a certain amount of food into the domain at random locations. Then we check for every agent if it has access to food. If it doesn't, it dies with probability d. If it has access to food, it reproduces with probability r, meaning that the individual creates an offspring (if there are more individuals than there is food available, the probability is lowered proportional to the food deficiency). Finally, all living individuals move randomly over the domain within a radius at most their motility value m (mathematically, a direction is chosen at random and than the step size is chosen uniformly randomly between 0 and m).

Evolution can be implemented in this system as follows. When an offspring is created, it will inherit the traits of their parent, but not precisely; a slight variation is applied on top of this, making it so that offspring can have slightly better or worse traits than their parent. Because individuals might die in this simulation, there will be selection based on individuals that are most adapt to the system (e.g., to the domain setup, competition with other agents, and food input).

Locate the files `evolution_IBM.ipynb` and `evolution_IBM_function.py` and put them in the same folder. Again only the Jupyter notebook needs to be opened and appended for this project.

1.1 Population model

First, run the code several times without evolution (e.g. keep the evolutionary parameters such that offspring will inherit precisely the traits of their parent). Look at the evolution of food and agents (make sure the `count_plot` and `domain_plot` are turned on). What do you observe? Play around a bit with the parameters m, d and r and see if and how the dynamics change. If you want, you can also change the food input over time by changing the function `food_input`.

TIP: Again, the visualisation is the most computationally expensive part of the code, so turning off the visualisations during the simulation does speed it up tremendously.

1.2 Parameter bounds

Often, not all values for parameters make sense and we need to impose sensible bounds on the parameters in a simulation. For this system, what are sensible bounds for the parameters m, d and r? In the code, it has been enforced that offspring have a motility m between 0 and 0.5, a death rate d between 0.01 and 1, and a growth rate r between 0 and 1. Do these bounds make sense to you?

1.3 Evolution

When you have some intuition for the system without evolution, see if things change when the evolutionary parameters are non-zero. A good starting point would be to set the variation in one of the parameters to something small like 0.01 (this is the maximal variation between the traits of the offspring and the parent). If you turn on `plot_evolution` you will be able to see the variation in parameters m, d and r during the simulation (but at the expensive of additional computation time for the simulation). Alternatively, you can use the last block of code in the main file to analyse the output after the simulation.

1.4 Additional suggestions

If time permits, continue exploring evolution in this model. For instance, you can turn on evolution for the other parameters or start with different parameters. Another possibility is to experiment with the food input function: what happens if the food input changes over time and more or less becomes available? Does this influence the evolution in the model? There is the option to make the food function dependent on the number of agents (`nr_agents`) to have an adaptive food input.