Creating a Forest-Fire Simulation using Cellular Automata in Python 3.9

A Brief Overview of Simulations and Cellular Automata

A simulation is used to model a real life system. There are two types of simulations: analytical and numerical. Analytical simulations require a high knowledge of math and produce precise but general analytical solutions for a real life problem. Numerical simulations use computers to crunch the math and produce numerical solutions for a real life problem that analytical simulations may find impossible or almost impossible to produce. The equation for a numerical simulation is: $state_t = rules(state_{t-1})$. This is a basic equation but is powerful nonetheless. The equation states that the current 'state' of the system is solely determined by applying a specific set of defined rules or equations to the directly previous 'state' of the system. Cellular automata is a special type of simulation which uses a grid of discrete cells. Each cell is assigned a starting value in the initial time step. Then each cell is updated to the next time step, via a fixed set of rules which are applied to the grid of cells. The key difference that separates cellular automata from a numerical simulation, is that the state of a cell's neighbourhood can impact the cell itself in the next time step. The forest-fire model described in this report is meant to simulate the cycle of forest fires moving through a continually growing forest. This simulation is not physically accurate, however it works well for showcasing cellular automata.

Part 1: The Forest-Fire Model

The forest-fire model that will be discussed in this report was created in JuptyerLab using python 3.9. The forest-fire model created for this project was based on cellular automata. The cells of the grid were programmed to be one of three states: an empty cell, a tree cell or a fire cell. Between each time step the cells were updated in accordance to the rules of the system. The rules of the system were: 1. A fire cell in the previous generation turned into an empty cell in this generation (burn out), 2. In this generation a tree cell turned into a fire cell, if one of its neighboring cells was a fire cell in the previous generation (fire spread), 3. A tree cell has a set probability *f,* of turning into a fire cell during this generation (lightning strike), 4. An empty cell has a set probability *g*, of turning into a tree cell during this generation (tree growth). If none of the rules can be applied to a cell then that cell remains unchanged from the previous generation. The von Neumann neighbourhood was used to define which cells were neighbours to one another and thus determine the impact each cell had on its neighbours. The von Neumann neighbourhood states that each cell was neighbours with the cells to their north, south, east and west.

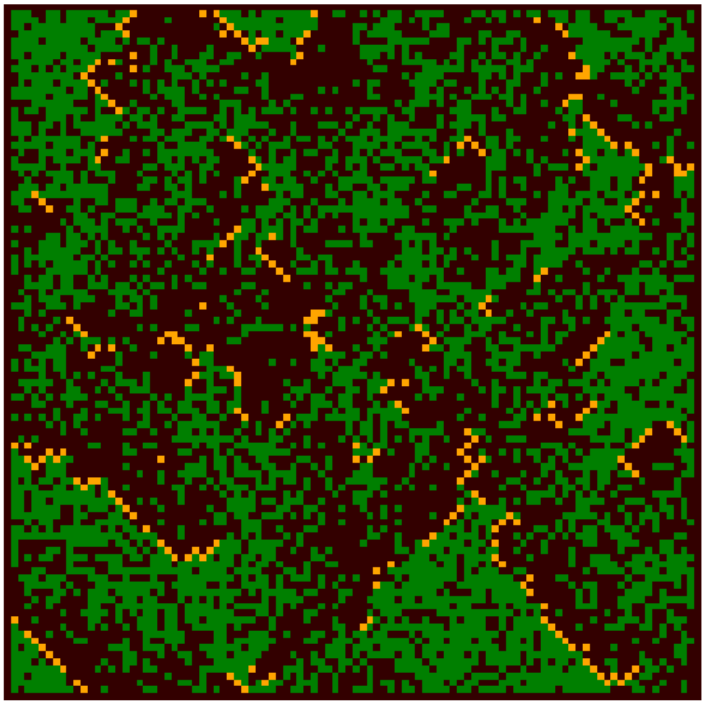

Fig. 1 A screenshot of the forest fire simulation, which was generated using matplotlib in python.

The Starting Grid, Probabilities and Edges

The grid size was set to 100 x 100 cells, so 10,000 cells in total. This was mainly because the animation of the forest-fire model looked aesthetically pleasing when the grid was set to these dimensions. To compensate for such a large grid the probabilities *g* and *f,* were set to 0.05 and 0.0001 respectively. These probabilities may seem quite low but there were 10,000 cells in the grid, so the number of 'lightning strikes' and 'trees growing' in each generation remained relatively consistent. The initial generation of cells were all set to empty cells and the 'forest' grew in over the first few generations. The tree growth probability was set high enough that it only took a few generations for tree cells to almost completely fill the grid. The cells that lined the edges of the grid were always set as empty cells and they did not interact with the cells within the grid.

Part 2: Steady-State Investigation

To get the simulation and animation to run effectively, a steady-state between the number of tree cells and the number of fire cells had to be reached. A steady-state was reached when the number of tree cells and the number of fire cells remained relatively consistent between each time step. The number of tree cells nor the number of fire cells that occupy the grid at any time cannot reach 100% or 0%, otherwise a steady-state would never be achieved. With the probabilities *g* and *f,* set to 0.05 and 0.0001 respectively a steady-state was achieved and can be seen in fig. 2. To investigate the relationship between tree growth (*g*), lightning strikes (*f*) and any other factors that may affect the steady-state, an analysis of the simulation was conducted.

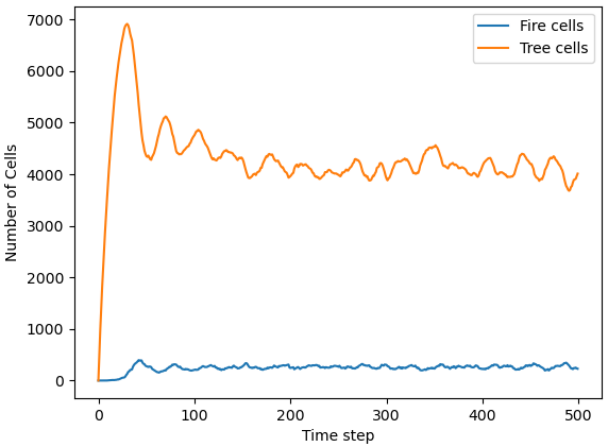

Fig. 2 A graph that shows the steady-state of fire cells and tree cells over 500 time steps. The probabilities *g* and *f,* were set to 0.05 and 0.0001 respectively. The number of fire cells and tree cells were both zero at time step one because all the cells in the initial grid were set to empty cells. After the first ~ 50 time steps a steady-state was achieved.

Analysis of the Steady-State

The first step was to see how tree growth (*g*) and lightning strikes (*f*) individually affected the number of fire cells and tree cells. The probability of tree growth was increased by 0.01 from 0.01 to 1. The simulation was ran for 100 time steps per probability and the mean number of fire cells and tree cells were recorded at each probability. The resulting plot from this analysis showed that as the probability of tree growth (*g*) increased from 0.01 to 1, the mean number of tree cells per simulation more than doubled from ~3500 cells to ~8500 cells (see fig. 3). The average number of fire cells increased from ~20 cells to ~1200 cells as the probability *g* increased from 0.01 to 0.4. The average number of fire cells then slightly decreased as the probability *g* increased from 0.4 to 1. However the average number of fire cells or tree cells was never recorded as zero for any of the 100 probabilities tested, so the number of tree cells and fire cells were always in a

steady-state. When the probability *g* was dropped below 0.01 the number of tree cells regenerating each time step was too small. Thus, the tree cell density was not high enough for fire cells to spread, so the tree cells and fire cells were not in a steady-state. The probability of lightning strikes was increased by 0.01 from 0.0001 to 0.01. The simulation was also ran for 100 time steps per probability and the mean number of fire cells and tree cells were recorded at each probability. Increasing the probability of lightning strikes decreased the average number of tree cells from ~5000 to ~3000 (see fig. 4). The average number of tree cells started to plateau as the probability increased to 1. The number of fire cells remained consistent and did not increase or decrease as the probability of lightning strikes increased. Again the average number of tree cells and fire cells never reached zero, so the number of tree cells and fire cells were always in a steady state. Overall it appeared that the probabilities *g* and *f* affected the average number of tree cells and fire cells but did not affect the steady-state of the system unless they were set incredibly low. Decreasing or increasing the grid dimensions whilst keeping the probabilities of *g* and *f* set at 0.05 and 0.0001 respectively did affect the steady-state of the system. The probabilities of *g* and *f* were specific to a grid of 100 x 100 cells, or a grid similar in size. If the grid dimensions were set much higher or much lower than 0.05 and 0.0001, then the chances of trees growing and lightning strikes occurring would be either too frequent or too infrequent for a steady-state to occur. Anyone who wishes to replicate this simulation with different grid dimensions should adjust their probabilities of *g* and *f* accordingly.

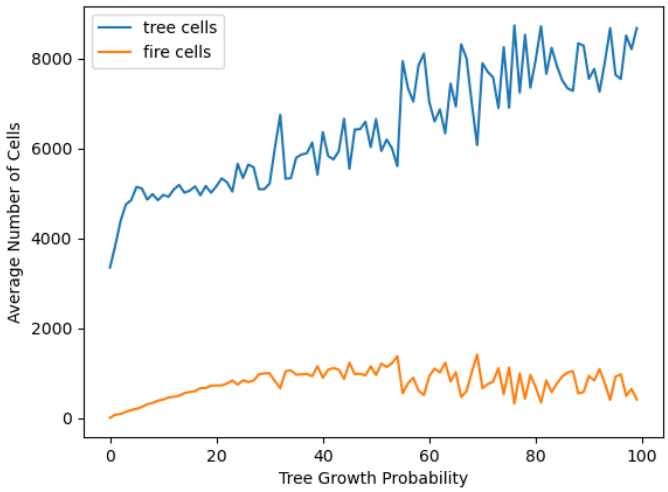

Fig. 3 The average number of tree cells increased as the tree growth probability increased. The average number of fire cells increased as the tree growth probability increased to 0.4 and then remained constant. The system remained in a steady-state at all probabilities tested.

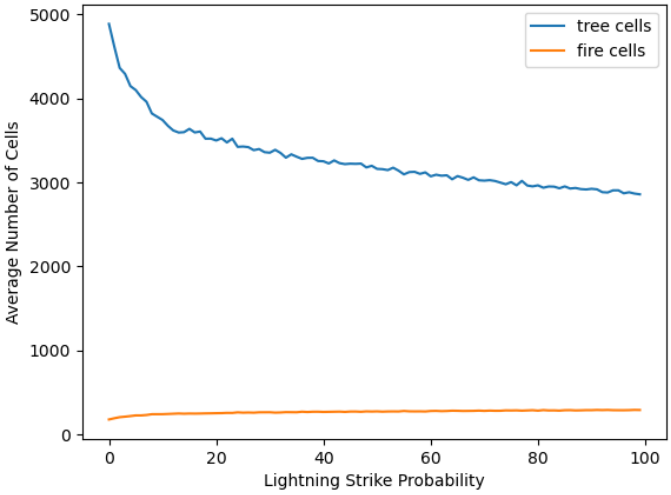

Fig. 4 The average number of tree cells decreased as the lightning strike probability increased. The average number of tree cells started to plateau as the probability increased. The average number of fire cells remained constant. The system remained in a steady-state at all probabilities tested.

Part 3: Updated Model

To update the forest-fire model, a different neighbourhood was added, changing the dynamics of the fire spread. The Moore neighbourhood was used instead of the von Neumann neighbourhood. Under the rules of the Moore neighbourhood, the fire can spread in four orthogonal directions (north, east, south, west) and four diagonal directions (north-east, north-

west, south-east, south-west). The fire spreads faster diagonally than it does orthogonally, so the probability of the fire spreading to diagonal cells was reduced to 0.6. Everything else in the model remained the same.

Impacts on The Steady-State

The steady-state remains intact using the updated model. It looks very similar to the steady state of the original model (see fig. 2 and fig. 5). The tree growth probability (*g*) and the lightning strike probability (*f*) were analysed to assess if the Moore neighbourhood affects the system and the steady-state. The average number of tree cells stayed consistently around ~4500 cells when the tree growth probability was increased from 0.01 to 1 (see fig. 6). However the fire cells gradually increased from ~20 cells to ~2500 cells. So the Moore neighbourhood had a differing effect on the number of fire cells and tree cells at differing probabilities of tree growth than the von Neumann neighbourhood did. The Moore neighbourhood increased the number of fire cells whilst the number of tree cells remained constant, whereas the von Neumann neighbourhood increased the number of tree cells whilst the number of fire cells remained fairly constant (see fig. 3 and fig. 6). The Moore neighbourhood and the von Neumann neighbourhood had a very similar effect on the average number of tree cells and fire cells as the lightning strike probability increased (see fig. 4 and fig. 7). The Moore neighbourhood just slightly decreased the maximum number of average tree cells but the trends in the data are exactly the same as the lightning strike probability increased. The system remained in a steady-state at all the probabilities tested.

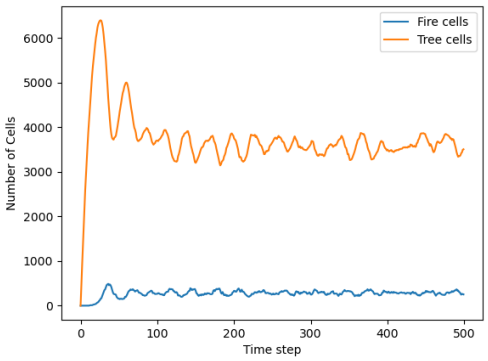

Fig. 5 The steady-state of the updated model. It looks very similar to the original model's steady-state (see fig. 3).

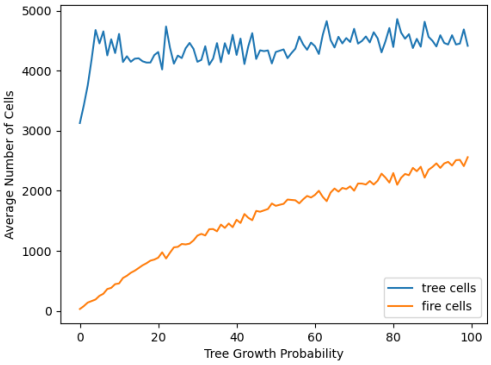

Fig. 6 The number of tree cells remained relatively consistent as the tree growth probability increased. The number of fire cells gradually increased as the tree growth probability increased. The system remained in a steady state at all probabilities tested.

Conclusion

This simulation of a forest fire is not physically accurate but it does a good job of showcasing cellular automata. The animation in particular looked dazzling to observe. Updating the model to add a more realistic fire spread mechanic did not affect the steady-state of the system but did affect the number of tree cells and fire cells. Updating the model with more realistic mechanics like wind, rain, atmospheric temperatures and different types of trees may lead to a more physically accurate simulation which could be used to model real forest fires. These physically accurate simulations could be used to test new methods of forest fire control and prevention tactics.

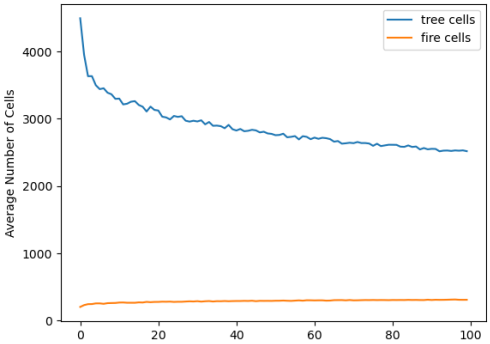

Fig. 7 Exactly the same trend as fig. 4. The number of fire cells remained constant and the number of tree cells plateau. The system remained in a steady-state at all probabilities.