

Day 18 Part 1 In-class Assignment: Modeling forest fires with an Agent-based Model

✔ Put your name here.

✔ Put your group member names here.



Goals of this assignment

The primary goal of this assignment is to model the spread of a forest fire using an agent-based model (ABM). In doing so, we will:

- Use ABM to model forest fires
- Examine and quantify the concept of a “tipping point” in a model.

Assignment instructions

Work with your group to complete this assignment. The first part of the assignment involves working out a plan for your model on the whiteboards and taking a picture of it. This picture should be uploaded to D2L along with your notebook in the appropriate submission folder. The assignment is due at the end of class.

Welcome to Part 1 of ICA 18!

We have split this assignment over two days, so that you can become more comfortable with material. On the first day, you will build the basics of the forest fire model and the second day you will analyze the model and the tipping point! Let’s get started!

Reviewing the motivation for the model

Why model forest fires?

While this isn’t a huge problem in Michigan, the *states in the western United States having been suffering a tremendous problem with huge and difficult-to-control forest fires*. This comes from a combination of extended drought conditions, dense woodlands, and forest management policies that suppress small fires and thus ensure that large quantities of dead, dry trees and brush are available to burn when a large fire inevitably starts (typically from lightning strikes, but occasionally from negligent campers, or recently from gender-reveal celebrations). In recent years, this has been exacerbated by climate change, which has both caused drought conditions to be more severe and allowed tree-killing diseases and insects to flourish, which produces more dead, easily-burned wood.

These forest fires destroy ecosystems and peoples’ homes and other property, and can result in the loss of human and animal life. A key challenge in forest management is to attempt to contain these huge forest fires once they start, in order to protect human lives, settlements, and infrastructure. To that end, it is critical to have models of how fire spreads in various conditions; see, for example, the [Open Wildland Fire Modeling group](#).

The Rules for Our Model

Setup

The model is a two-dimensional square $N \times N$ array that represents the forested region we’re simulating. **The cells in the array can have three values:**

- 0 (empty)

☰ Contents

✔ Put your name here. [Print to PDF](#) ▶

✔ Put your group member names here.

[Goals of this assignment](#)

[Assignment instructions](#)

[Welcome to Part 1 of ICA 18!](#)

[Reviewing the motivation for the model](#)

[Why model forest fires?](#)

[The Rules for Our Model](#)

[Setup](#)

[Neighborhoods](#)

[Evolving with Time](#)

[Ending Conditions](#)

[Your mission at the end of the entire assignment:](#)

[Part 1: Planning your solution](#)

[Part 2: Implementing your solution](#)

[2.1 First important Function: Plotting the grid!](#)

[2.2 Initializing the forest](#)

[Part 3: The main event: Make the Fire Spread \(evolving over time\)!](#)

✔ [3.1 In this model, the main agent you have to deal with is the fire. Your job is to write the function that controls how the fire moves.](#)

✔ [3.2 As good coders, we always test our new functions! Test your function above and make sure it works!](#)

✔ [3.3 Preparing for Next Class](#)

[Congratulations, you’re done!](#)

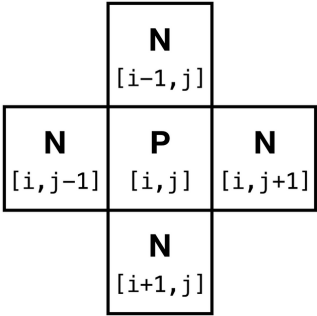
- 1 (trees)
- 2 (on fire)

At the beginning of the model, a user-specified fraction of the cells $f_{\text{trees_start}}$ are randomly filled with trees and the remaining cells are empty.

One edge of the board (say, the entire leftmost column) is set on fire.

Neighborhoods

Each cell has a “neighborhood” that is composed of its four neighbors to the left, right, above, and below it. **If a cell is along one of the edges of the array, only consider the neighbors that it has** (i.e., don’t try to go out of the bounds of the array).



Evolving with Time

The model takes steps forward in time, where every cell is modified based on the previous step. The model evolves as follows:

- If the cell was empty last turn, it stays empty this turn.
- If the cell is a tree and any of its neighbors were on fire last turn, it catches on fire.
- If the cell was on fire last turn, the fire has consumed all of the trees and it is now empty.

Ending Conditions

The model evolves forward in time until all of the fires have burned out.

After this happens, you can calculate the fraction of the cells that still have trees at the end of the model ($f_{\text{trees_end}}$) and the fraction of cells that are empty (f_{empty}). The fraction of burned cells, f_{burned} , is just the difference between the fraction of cells that were initially trees and the fraction of cells that are trees at the end of the model; in other words,

$$f_{\text{burned}} = f_{\text{trees_start}} - f_{\text{trees_end}}$$

Your mission at the end of the entire assignment:

Your mission is to answer the question: “How does the spread of fire relate to the density of the forest?”
More precisely, we’re asking “How does f_{burned} depend on $f_{\text{trees_start}}$?”

To achieve this mission, we will break this down into a few parts:

Part 1: Planning your solution

✔ As a group, create a plan to implement this model for an arbitrary value of $f_{\text{trees_start}}$. Make sure that you think about how to set up the initial conditions, how to evolve the model, and how to calculate the fraction of trees and empty cells that remain in the end. What will the board look like to start? What will it look like in the end? How will the forest fire move?

Important: Make sure you discuss how you will handle cells that are on the boundaries of your 2D board!

Make sure to chat with an instructor after you have completed this step. Do not spend more than 20 minutes on this part of the activity!

 Feel free to use the whiteboard or put some thoughts here

Part 2: Implementing your solution

Now we’re going to work through a combination of provided code and code that you have to write. The goal is to have all the pieces to build the forest fire model by the end of class!

Make sure to execute the following cell of imports before you move on!

```
# standard includes
import numpy as np
import numpy.random as rand
%matplotlib inline
import matplotlib.pyplot as plt

# Next we are going to import some specific libraries we will use to get the animation
to work cleanly
from IPython.display import display, clear_output
import time
```

2.1 First important Function: Plotting the grid!

Take a look at the `plotgrid` function. You were given a similar one in your pre-class assignment. We'll be using this code a lot for displaying your forest, so we want to make sure you understand it.

✅ Complete the code below, by filling in the “___” spots, and add comments to explain what each line of the code is doing.

```
# Function plotgrid() does what??

def plotgrid(myarray):

    #
    x_range = np.linspace(0, myarray.shape[1]-1, myarray.shape[1])
    y_range = np.linspace(0, _____) # repeat for the y/vertical axis

    #
    x_indices, y_indices = np.meshgrid(x_range, y_range)

    #
    tree_x = x_indices[myarray == 1];
    tree_y = y_indices[_____];
    fire_x = x_indices[myarray == 2];
    fire_y = y_indices[_____];

    #
    plt.plot(tree_x, myarray.shape[0] - tree_y - 1, 'gs', markersize=10)
    plt.plot(_____) # repeat for indices with fire

    #
    plt.xlim([-1, myarray.shape[1]])
    plt.ylim([-1, myarray.shape[0]])

    #
    plt.tick_params(axis='both', which='both',
                    bottom=False, top=False, left=False, right=False,
                    labelbottom=False, labelleft=False)
```

2.2 Initializing the forest

Before you can run a forest model, you need to initialize the board. You should have already done this in your pre-class assignment, so we're providing the following code for you.

✅ Take a look at it and make sure you understand it. How does it compare to the code that you wrote?

```
def set_board(board_size=50, f_trees_start=0.5):
    '''
    Creates the initial game board.

    Inputs:
        board_size: length of one edge of the board
        f_trees_start: probability that a given cell is a tree
                      (effectively the tree density)

    Outputs a 2D numpy array with values set to either 0, 1, or 2
    (empty, tree, or fire)
    '''

    # all cells initialized to 'empty' (0) by default
    game_board = np.zeros((board_size, board_size), dtype='int64')

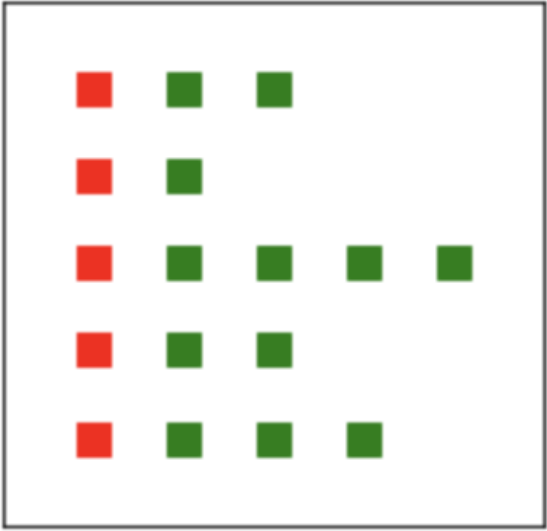
    # loop over board and roll the dice; if the random number is less
    # than f_trees_start, make it a tree.
    for i in range(board_size):
        for j in range(board_size):
            if rand.random() <= f_trees_start:
                game_board[i, j] = 1

    # set the whole left edge of the board on fire. We're arsonists!
    game_board[:, 0] = 2

    return game_board
```

2.3 Testing your Code

Execute the cell below to make sure that your `plotgrid()` and `set_board()` functions are working correctly. Your output should match this picture.



```
# run this cell
rand.seed(1234)
fig = plt.figure(figsize=(3,3))
small_board = set_board(board_size=5, f_trees_start = 0.75)
plotgrid(small_board)
```

✔ 2.3 Exploring the `set_board()` and `plotgrid()`!

Now, create a board using the default arguments for the `set_board()` function and plot the board with `plotgrid()`. Once you have done that, answer these questions.

- Does the fire show up in the places you want?
- Does the tree fraction or board size change accordingly when you change the input parameters?
- What happens if you make the board size much larger than the default?

Put your answer here

```
# put your test code here. Make a plot to check it.
# we're going to define a default figure size for you to make things look nice
fig = plt.figure(figsize=(10,10))
```

✔ **Task:** Explain the process of initializing the forest to someone who has never coded before. What are the rules? What do we hope the board looks like at the start?

Put your answer here

Part 3: The main event: Make the Fire Spread (evolving over time)!

Clearly the most import part of an agent-based model is figuring out how your agents should behave in your simulation.

✔ 3.1 In this model, the main agent you have to deal with is the fire. Your job is to write the function that controls how the fire moves.

The skeleton function provided below:

- Takes in the *current* board,
- Defines a *new* board, and then
- Returns that *new* board where the positions of the fire have been updated.

Make sure you are updating the values on the **new board**, based on the values of the **current board**! This function on it’s own is only responsible for one step in time. It takes in the current state of the board and based on that state, defines the next state of the board.

To avoid needless scrolling **here are the rules of the model again:**

Each cell has a “neighborhood” that is composed of its four neighbors to the left, right, above, and below it. (Note: not the diagonal elements!) If a cell is along one of the edges of the array, only consider the neighbors that it has, and don’t try to go out of the bounds of the array!

The model takes steps forward in time, where every cell is modified based on the previous step. The model evolves as follows:

- If the cell was empty last turn, it stays empty this turn.
- If the cell is a tree and any of its neighbors were on fire last turn, it catches on fire.
- If the cell was on fire last turn, the fire has consumed all of the trees and it is now empty.

The function contains comments to suggest the steps that you need to implement to make it work. Think about how the board is set up and how to access the elements within the board.

The functions `OnBoard` and `getNeighborValues` from last class will be very helpful for this function!

```
def advance_board(game_board):  
    '''  
    Advances the game board using the given rules.  
    Input: the initial game board.  
    Output: the advanced game board  
    '''  
  
    # create a new array that's just like the original one, but initially set to all  
    # zeros (i.e., totally empty)  
    new_board = np.zeros_like(game_board)  
  
    # loop over each cell in the board and decide what to do.  
    # You'll need two loops here, one nested inside the other.  
  
    # Now that we're inside the loops we need to apply our rules  
  
    # if the cell was empty last turn, it's still empty.  
    # if it was on fire last turn, it's now empty.  
  
    # now, if there is a tree in the cell, we have to decide what to do  
  
    # initially make the cell a tree in the new board  
  
    # If one of the neighboring cells was on fire last turn,  
    # this cell is now on fire!  
    # (make sure you account for whether or not you're on the edge!)  
  
    # return the new board  
    return new_board
```

✔ 3.2 As good coders, we always test our new functions! Test your function above and make sure it works!

Run the code a few times to see whether fire advances according to the rules. You can start with any sort of initial board that you want, you can even just use the default board for the purpose of testing. Remember the advance_board function only advances one step in time.


```
# Initialize a new board here  
  
# Test your new advance_board function here. Make a plot to check it.  
# Run this cell several times to make sure the fire continues to advance.  
  
# Again, we set the figure size for you  
fig = plt.figure(figsize=(10,10))
```

✔ 3.3 Preparing for Next Class

Great work! You now have code that begins to model a forest fire! Next class, we are going to use the advance_board function to complete the forest fire model and incorporate the ending conditions. We are also going to analyze the tipping point of the model.

To prepare us for next class, discuss these questions with your group and write down some of your thoughts:

- How is the advance_board function going to help you model the forest fire? What if you wanted to advance the board 10 steps? Is there an easy way to that?
- When is the forest fire over? What would the board look like? What might that look like in code?

 Put your answer here

Congratulations, you’re done!

Submit this assignment by uploading your notebook and the pictures of your model planning session to the course Desire2Learn web page. Go to the “In-Class Assignments” folder, find the appropriate submission link, and upload everything there. Make sure your name is on it!

© Copyright 2018, Michigan State University Board of Trustees

By MSU CMSE
© Copyright 2021.