Modelling of forest fires

Year 2023 – 2024

Andrei Burca, Matei Barbat, Isaac Michiels, Caparros Hugo -10th Grade
Stefan Victor Oprean, Sofian Neagu, Stefan Martin - 11th Grade

Schools : Colegiul National "Emil Racovita" Cluj-Napoca, Lycée Val de Durance Pertuis
Teachers : Ariana-Stanca Văcăreţu, Hubert Proal
Researcher : Yves Papegay, INRIA - Sophia Antipolis Méditerranée Research Center.

Contents

1. Research Topic

Using automated cells, the forest is represented by:

- Green cells: Trees
- White cells: Nothing
- Grey cells: Ashes
- Red cells: Fire

Process:

1.Ignition: We light up a cell (red).
2.Fire Spread: If a tree (green cell) is adjacent to the ignited cell (above, below, left, or right), it catches fire (red).
3.Burning: If a tree is on fire, in the next second, it becomes ashes (grey cell).
4.Aftermath: If a fire is in ashes, in the next second, it turns white, indicating there is nothing further to burn (white cell).

Objective:

To understand and formalize the model of cellular automata for this process and, if possible, carry out simulations. What can be concluded?

2. The Real Life Problem

In today's world, forest fires have become a global threat to both animals and humans. These fires are occurring more frequently and are increasingly difficult to contain. Due to global warming, firefighters struggle to keep up with the rapid expansion of forest fires. Although they employ sophisticated techniques to halt the spread of fires, these methods often demand substantial resources and physical strength, sometimes proving insufficient.

3. Objective

Our objective is to create a model of cellular automata and conduct simulations based on simple rules of fire expansion. By doing so, we aim to identify a distribution of trees and gather useful information that can be applied in real-life scenarios. Discovering specific patterns will help balance fire expansion and the preservation of trees, ultimately providing firefighters with a more efficient and effective strategy.

4. Approaches and Problem Solving

Each team approached the problem from a unique perspective, bringing different visions and strategies for solving this issue.

4.1 First Approach:

We decided that our initial step must be to understand fire dynamics comprehensively. We gathered information about real-life forest fires and their mitigation strategies.

Subsequently, we recognized the necessity of conducting simulations on paper to observe fire spread patterns. Our first conclusion was that when igniting a fire from a specific position, it is essential to isolate the fire and protect as many trees as possible. The spread of fire resembles a chain reaction.

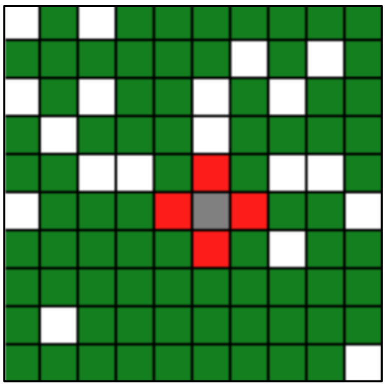

Figure 1- The start of a chain reaction

Given the research topic and its guidelines, we determined that the fire should not have any trees in adjacent positions. However, we realized that it is impossible to predict the fire's ignition point. Therefore, to obtain valuable information, we needed to conduct simulations starting from **every cell** containing a **tree**. We also discovered that the most crucial data from a simulation are the initial percentage of trees (out of 100%) and the final percentage of trees remaining.

For example:

- A green cell represents a tree.
- A white cell represents an empty space.

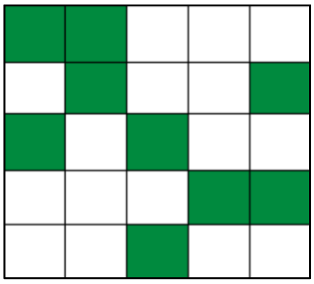

Figure 2 - Example of a matrix

For this matrix, we must conduct a simulation starting from each cell that contains a tree. There are nine trees, which means nine simulations are required. The first fire will start from the tree in the top-left corner and continue until it extinguishes. This will result in a matrix that illustrates the final percentage of trees remaining. We will then reset the matrix to its initial state, with no trees burned, and proceed to the next tree from which the fire has not yet started.

After completing these simulations, we will have the initial and final percentages of trees from each simulation. With this data, we will create a new variable called the "**Average Final Percentage**," defined as the sum of all final percentages divided by the number of simulations. The Average Final Percentage is the main indicator that shows us if a matrix is actually useful and protects many trees. How do we compare two matrices? We just compare their Average Final Percentages and the one that has a higher Average Final Percentage is the better one. [1]

Now, we wanted to see what would happen if there wouldn't be any trees neighbouring each other so that we could preserve as many trees as possible.

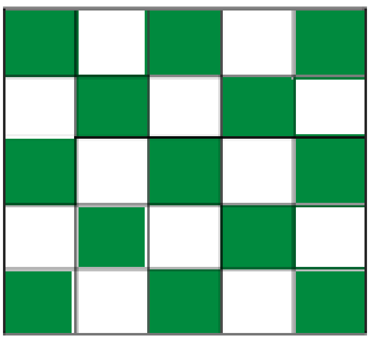

Figure 3 - A matrix that has no neighboring trees

In this case we found that we will have an Average Final Percentage that is approximately 50% (based on the dimension of the matrix the Average Final Percentage can be less or more than 50%). For every simulation concluded we will only have one tree that will burn, the one from which the fire has begun. This is a starting point because it demonstrates that we don't need to carry out simulations for matrices that have an initial percentage less than 50% because we already have a matrix that has an Average Final Percentage higher.

We have decided that now we know enough information about our research problem to start using computer simulations that could speed up the process.

We wrote a simple program in C++ where we can input a certain matrix and give its parameters or can generate a random matrix and in both cases it will have as output the Average Final Percentage.

1. https://github.com/MateInfinit/masud_code/blob/main/main.cpp

This C++ code simulates the spread of fire in a forest using a grid and records statistics about the spread. It initializes the grid with random values representing trees and empty spaces. The randnum function sets up the grid with trees (2). The **simulare** function simulates fire spreading from each tree cell, updating the grid over time until no more cells can catch fire. The main function runs the simulation once, initializes the grid, prints the initial state, and runs the fire spread simulation for all tree cells. Finally, it calculates and prints the percentages of trees before and after the fire and the time taken for the fire to spread. The results are printed to an output file.

2. https://github.com/MateInfinit/masud_code/blob/main/ForestFireSimulationGUI.py

This Python code uses the Tkinter library to create a graphical simulation of a forest fire. The code consists of two main classes: **'ForestFireSimulation'** and **'ForestFireGUI**'.

'ForestFireSimulation' initializes a grid where cells can be either trees (green), burning trees (red), burned trees (grey), or empty spaces (white). The simulation starts with a specified cell on fire. The **'update'** method progresses the fire by changing the state of cells based on their neighbors. Cells next to a burning tree catch fire, burning trees become burned, and burned trees turn into empty spaces.

'ForestFireGUI' creates the user interface for inputting simulation parameters such as grid width, height, tree density, and the starting position of the fire. When the "Run Simulation" button is pressed, it starts the simulation.

'ForestFireSimulation' runs the simulation, displaying the grid on a canvas and updating it every 2 seconds.

The main function initializes the GUI and starts the Tkinter event loop. The simulation updates and redraws the grid, showing the spread of fire visually.

4.2 Second approach

The second approach we took to obtain the needed data was to create a fine-tuneable program to simulate forest fires instead of manually simulating the problem or using real forests. The program was developed using Typescript (with Vite and D3) offering faster development and cross platform capabilities. You can find links to the working web app and the code below:

Web App	Code
https://forest-fire-simulation.web.app/simulation	https://github.com/game-geek/forest-fire-

4.3 Third approach

This code uses the Lee algorithm to simulate the spread of fire on a grid, similar to the first code but using breadth-first search (BFS). It is a research-based approach to model such phenomena.

3.https://github.com/MateInfinit/masud_code/blob/main/Lee

Explanation:

1.Initialization: Reads grid dimensions and initializes the grid from the input file. Identifies the starting point of the fire (2).

2.Lee Algorithm: Uses BFS to simulate fire spread:

- Starts from the initial burning cell.
- For each cell, checks its four neighbors (left, down, right, up).
- If a neighbor is a tree (1), it marks it as burning (-1) and adds it to the queue.

3.Output: Writes the final grid state to the output file.

This approach uses BFS to simulate fire spread efficiently, illustrating an alternative method to achieve the same results as the first code, suitable for a research paper on forest fire simulation algorithms.

5. Solution

In order to find the best matrix for a certain dimension we needed to find the matrix that has the highest Average Final Percentage (54 % Average Final Percentage). We decided that we needed to focus only on a certain dimension of the matrix because every dimension has another solution. We have chosen to present the solution for the 5 x 5 matrix. We have chosen this dimension because the solution presents a symmetric matrix.

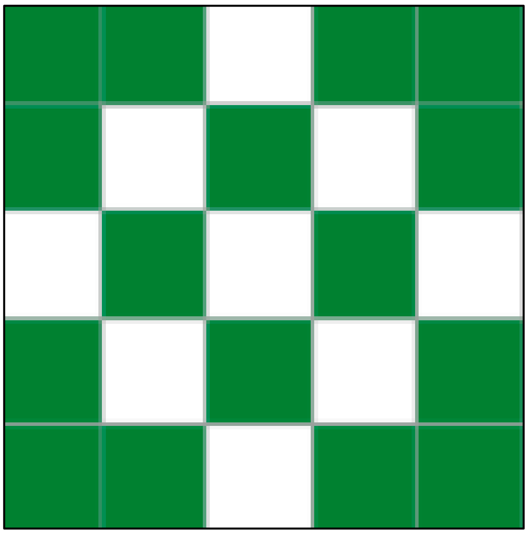

Figure 4 - Simple Model

If we compare this model with matrices with the same dimension, we will see that it has the highest Average Final Percentage. We have also verified that this matrix has the highest Average Final Percentage by making a program that calculates every Average Final Percentage of every matrix that is 5 x 5.

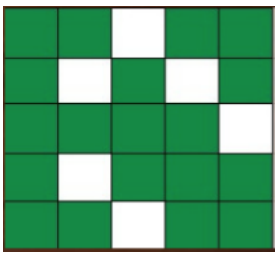

Figure 5 - Model 2

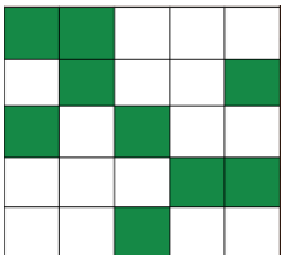

Figure 6 - Model 3

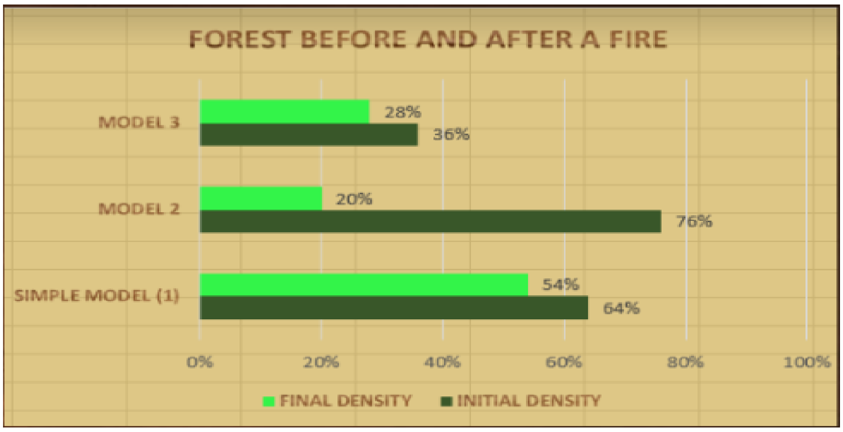

Figure 7 - Comparison between the three matrices

In order to maximize the potential and the efficiency of this matrix, we had to find a symmetrical pattern. [2] By doing so, we managed to expand this matrix and re-use the pattern in bigger matrices.

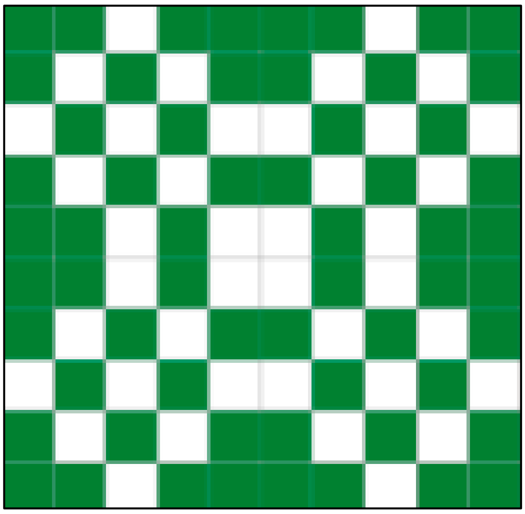

Figure 8 - Extended version of the matrix

6. Mockup

Using different materials, we managed to create a mockup which represents a 5 on 5 matrix. After making the base, we constructed little trees made out of toothpicks and artificial moss in order to pin them down on the cardboard base.

Our idea was to transport the physical representation of a matrix into a digital environment in order to create all different kinds of matrices and to make the topic and our project more interactive. By doing so, we came to the conclusion that we can include other students in our experiment by allowing them to arrange the trees in order to make certain patterns.


Figure 9 and 10 - Some of our classmates' matrices

The process of transporting the physical form of the matrix into the digital one required us to use an Image Recognition software, also known as an API. We chose the API from AWS Rekognition in order to analyse the matrices made in real life.

Figure 11 - MATh.en.JEANS Congress, Avignon Presentation

First, we arranged the trees in the desired pattern. Second, we took a photo of the whole matrix in order to introduce it in the Image Recognition software. After we completed these two steps, we moved the image into the folder used by the software. From there, the software separated the image into smaller images, each representing a matrix cell, placing each cell into the correct position in the digital matrix. The last step was for us to run the code which analyses the matrix and returns us the Average Final Percentage.

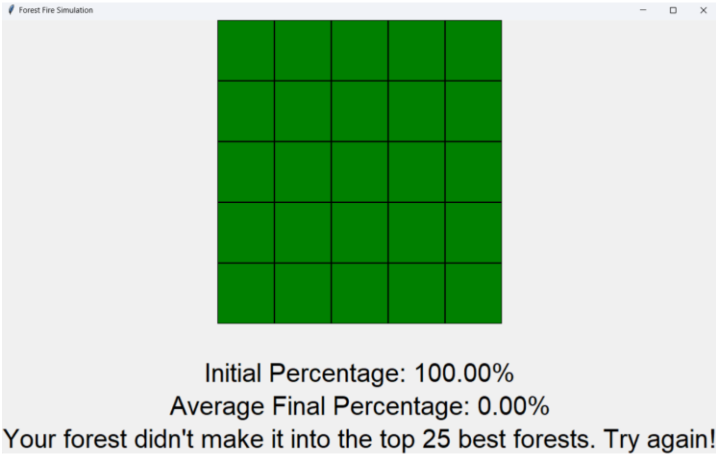

Figure 12 - The result of a simulation

7. Conclusion

In conclusion, our research focuses mostly on the amount of trees remaining (Average Final Percentage). There are a lot of variables still left to explore, such as weather (for example wind or rain) or possibility of the expansion of the fire (how often does the fire spread to the cells around it) which we will explore in order to find a much more complex and complete solution to the problem. These would help us have a better understanding of the subject at hand and come to a much more complex solution to the problem.

EDITION NOTES

[1] It makes sense to consider the Average Final Percentage as a criterion for comparing two matrices. It might also be interesting to explore whether there is any relationship between this percentage and the initial one. This is partially done in Section 5 (Figure 7), but perhaps a deeper investigation could provide more information.

[2] It is indeed intuitive that symmetric matrices should be considered, but this fact deserves some explanation.