Module Number: 700109
Module Title: Advanced Computational Science
Title of Assessment: Modelling, Simulation and Report
Name: Balarabe Abubakar Tahir

Contents

List of Figures

List of Tables

BACKGROUND OF STUDY

Environmental damage, loss of property, loss of life and means of livelihood are some of the consequences or effects of a fire incidence. While this occurrence would cause more damage in urban areas, more interior or rural areas with forests are susceptible to the spread of fire rapidly and requires intervention to reduce the effect of potential destruction. Wildfires can occur anytime, anywhere, and are frequently brought on by human action or a natural occurrence like lightning. These reasons and more necessitates the scientific study of the spread of fire. Mathematical models are an essential component of fire mitigation. This is so because they are created and simulated to model fire behavior. They are also created and used to predict where a fire is likely to start, how rapidly it will spread (and in which direction), and how much heat it will produce. The importance of this study cannot be overemphasized as it leads to the safety of lives and significantly reduce financial losses.

INTRODUCTION

This report uses computational models like cellular automata and processing methods like sequential and parallel to model and simulate the development of fire in a forest.

A discrete computational model called a cellular automaton can be used to illustrate how components of a system interact with one another. In this research, cellular automata are used to mimic how a forest fire spreads on a grid employing sequential and parallel processing methods. The grid depicts a forest and could include a burning tree, a tree that isn't burning, or an empty space.

The aim is to demonstrate how the fire can spread in the grid using two distinct neighborhood approaches starting from this baseline grid. The Von Neuman neighborhood or Moore neighborhood are the choices for the neighborhoods. However, as instructed for this coursework, Moore's neighborhood was used for both the sequential and parallel implementation.

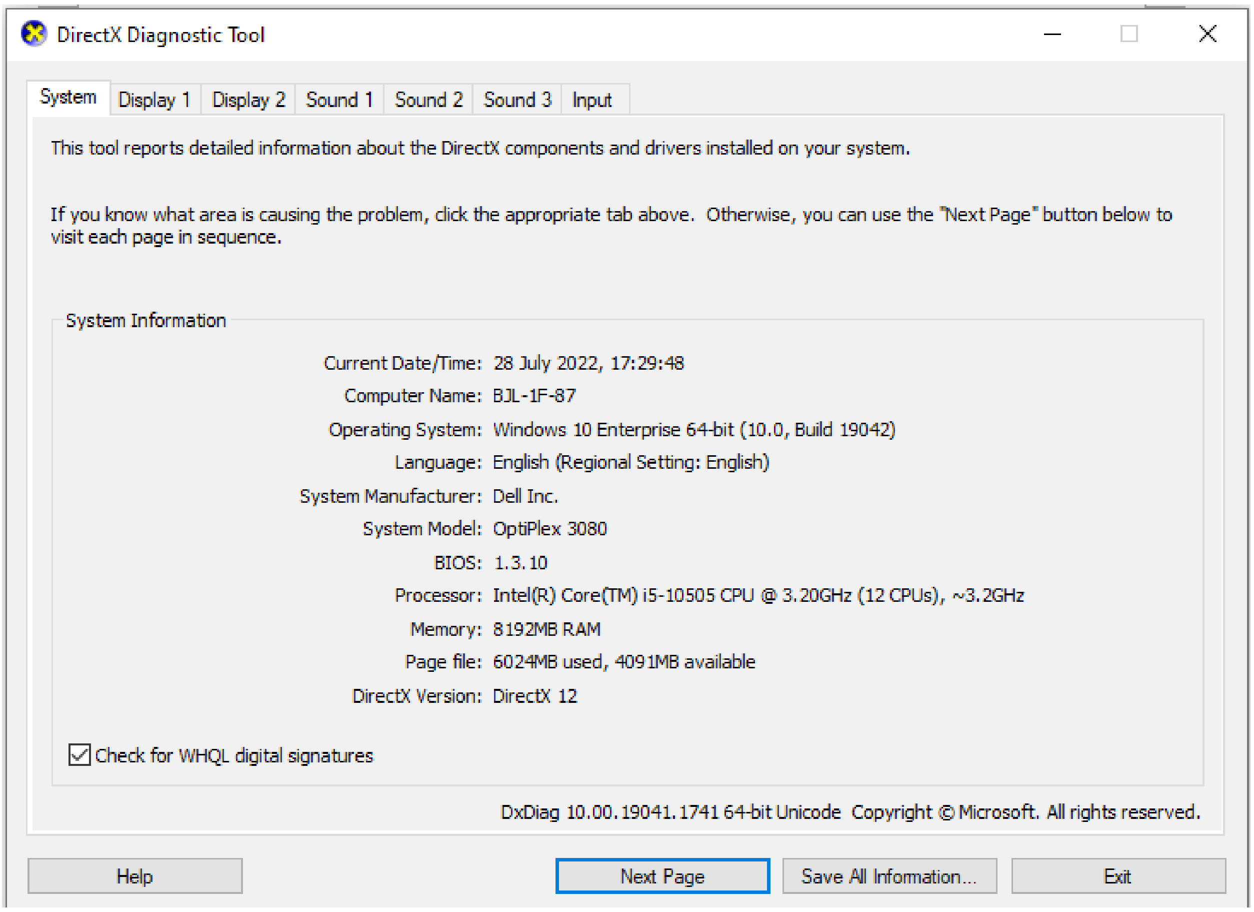
The following are potential learning outcomes from this project:

- Use and apply mathematical modeling in real-world simulations.
- To handle problems involving simulation and modelling, choose the proper computing architectures and methodologies.
- Apply cutting-edge scientific techniques to a challenging issue.

SYSTEM SPECIFICATION

The processing, storage, and memory requirements of a simulation must be supported by hardware. Certain simulations can grow noticeably sophisticated.

Mathematical rules must be able to be represented, processed, and processed under various programmed situations by software. A computer should generally have sufficient CPU, memory, and graphics cards (to process visual information).



This tool reports detailed information about the DirectX components and drivers installed on your system.

If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.

System Information

Current Date/Time:	28 July 2022, 17:29:48
Computer Name:	BJL-1F-87
Operating System:	Windows 10 Enterprise 64-bit (10.0, Build 19042)
Language:	English (Regional Setting: English)
System Manufacturer:	Dell Inc.
System Model:	OptiPlex 3080
BIOS:	1.3.10
Processor:	Intel(R) Core(TM) i5-10505 CPU @ 3.20GHz (12 CPUs), ~3.2GHz
Memory:	8192MB RAM
Page file:	6024MB used, 4091MB available
DirectX Version:	DirectX 12

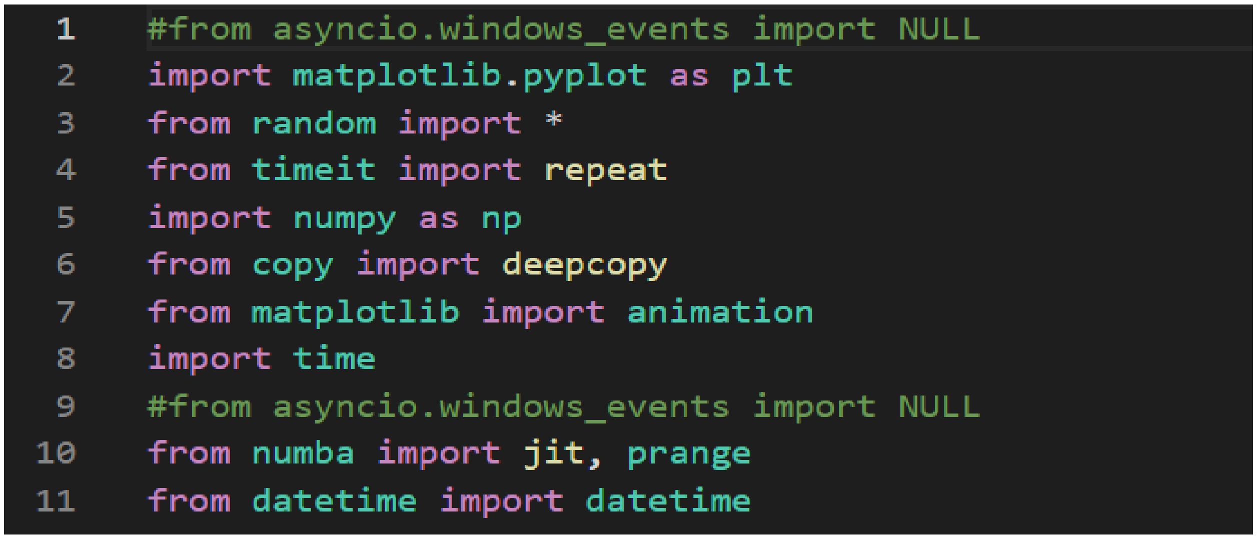
[✓] Check for WHQL digital signatures

DxDiag 10.00.19041.1741 64-bit Unicode Copyright © Microsoft. All rights reserved.

Fig1.0 System Specification

MODELLING

Python programming language and its dependencies was used in the implementation of this coursework. **Matplotlib** library was used for the interactive visual representation of the spread of fire. **Timeit** function was used to accurately measure the time taken for the spread of fire across the different grids specified in this coursework ranging from 400x400 to 2000x2000 for both parallel and sequential implementation of the moore's neighbourhood method. To create the arrays and matrices representing each block of the grid, **numpy** library was used. For the parallel implementation of fire spread, **numba** was used for fast compilation of and execution of the python code for simulation. A few other functions such as random, time and datetime was also called to give contextual details to this implementation.

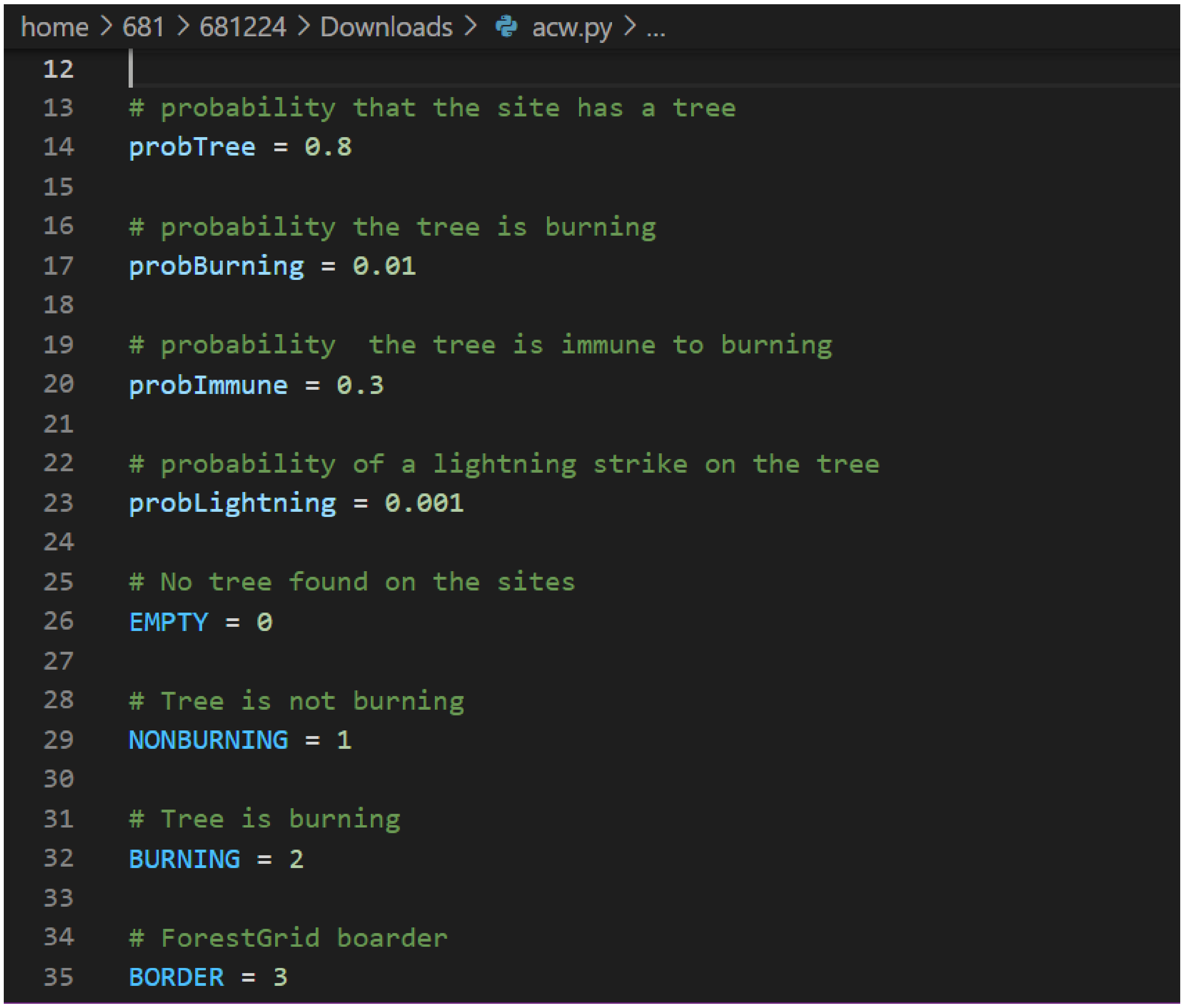


```python
#from asyncio.windows_events import NULL
import matplotlib.pyplot as plt
from random import *
from timeit import repeat
import numpy as np
from copy import deepcopy
from matplotlib import animation
import time
#from asyncio.windows_events import NULL
from numba import jit, prange
from datetime import datetime
```

Fig 2.0 Modelling

ASSUMPTIONS STATED

In creating this simulation, certain assumptions are stated (see snippet below).



```
home > 681 > 681224 > Downloads > acw.py > ...

# probability that the site has a tree
probTree = 0.8

# probability the tree is burning
probBurning = 0.01

# probability  the tree is immune to burning
probImmune = 0.3

# probability of a lightning strike on the tree
probLightning = 0.001

# No tree found on the sites
EMPTY = 0

# Tree is not burning
NONBURNING = 1

# Tree is burning
BURNING = 2

# ForestGrid boarder
BORDER = 3
```

Fig 3.0 Assumption Stated

Considering a typical forest site on fire, not all trees get burnt even though they are close to burning trees. It is also possible that a particular area has no tree, hence the site is empty. There is a probability that a natural occurrence of lightning starts the fire and other possible assumptions and probabilities. This is taken into account as seen from the image above. Only predetermined values that have been designated as constants may be present. The variables and their associated values are

- BORDER = 3
- EMPTY = 0
- NONBURNING = 1

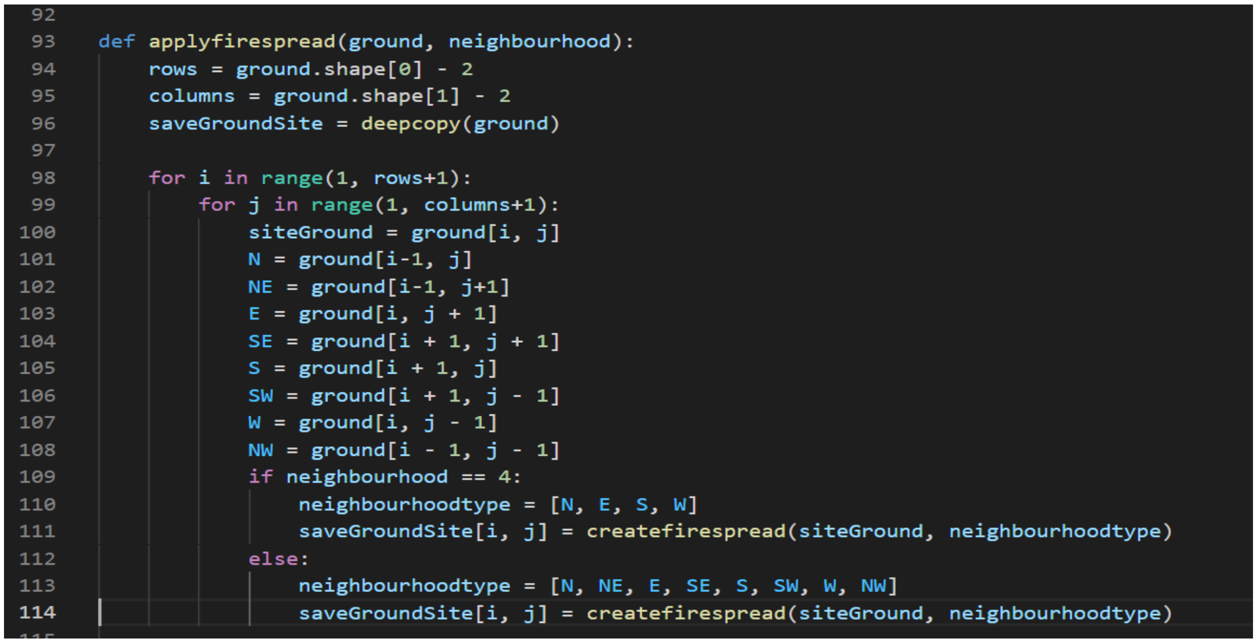
- BURNING = 2

Based on probability, these values are allocated to a site. The value of a given site is established by comparing specified probability values—also known as percentage chances of happening a random event. The following is a list of the predefined probability variables.

- probTree = 0.8
- probBurning = 0.01
- probImmune = 0.3
- probLightning = 0.001

The **createforestgrid** function is in charge of creating a grid and filling it with the site values (given a grid size of n). In order to calculate the value of a sites probability, we produce a random value between 0 and 1 using the Python Random Library. The end output of this function is a numpy array, which serves as both the initial grid and the starting point for the simulation of a forest fire.

Depending on its current value or the value of its neighbor, the **createfirespread** function determines a site's value for the subsequent iteration of the timestep. Either of the two approaches can be used to identify the neighbors. The techniques are von neumann which considers only East, West, North, and South directions while moore considers 8 directions which are taken into account. They are Northeast, Northwest, Southeast, Southwest, North, East, South, and West. As shown below, Moore neighborhood method used for applying the spread of fire.



```

def applyfirespread(ground, neighbourhood):
    rows = ground.shape[0] - 2
    columns = ground.shape[1] - 2
    saveGroundSite = deepcopy(ground)

    for i in range(1, rows+1):
        for j in range(1, columns+1):
            siteGround = ground[i, j]
            N = ground[i-1, j]
            NE = ground[i-1, j+1]
            E = ground[i, j + 1]
            SE = ground[i + 1, j + 1]
            S = ground[i + 1, j]
            SW = ground[i + 1, j - 1]
            W = ground[i, j - 1]
            NW = ground[i - 1, j - 1]
            if neighbourhood == 4:
                neighbourhoodtype = [N, E, S, W]
                saveGroundSite[i, j] = createfirespread(siteGround, neighbourhoodtype)
            else:
                neighbourhoodtype = [N, NE, E, SE, S, SW, W, NW]
                saveGroundSite[i, j] = createfirespread(siteGround, neighbourhoodtype)
```

Fig 4.0 Create Forest Grid

The **startfiresimulation** function is the primary function that accounts for all these earlier functions in the actual simulation section. It constructs a starting grid and executes the **applyfirespread** function at times in a loop, depending on the processing type and neighborhood method. When all is said and done, it outputs a collection of grids that can be used for visualization. The matplotlib library is used to create the visualization.

METHODS AND IMPLEMENTATION

The preceding functions represent a sequential approach to simulating and modelling the spread of fire in the forest. It is advisable to use sequential processing when there aren't many repeating jobs. But as we can see, the functions contain a lot of for loops and repeated activities. Therefore, a parallel processing technique that allows many CPU cores to be used simultaneously and divides iterative tasks to execute concurrently on multiple cores can be a superior solution.

By handling the division and communication of processes on an existing sequential algorithm, the Numba library aids in automating the parallel execution of a Python application. This can be achieved by using the @JIT annotation to parallelize a function, this was implemented in the project.

A pictorial visualization of the fire spread for different grids is depicted below. The animation function of the matplotlib library made this possible. The following images shows the fire spread through the saturation of the colors. Fig. 1 shows the start of the fire when the forest is still dense and full of trees. As the fire spreads, the effect is seen as shown in fig. 2 as many trees are randomly burning. The intensity of the burning is further shown in fig. 3 where many cells which once had trees on them are now empty as a result of the fire spread. Lastly, fig. 4 the full impact of the wildfire as it has spread across all the boundaries or grids where the fire occurred, leaving only the immune trees in scanty cells and majority of the other trees are completely burnt. This simulation was done for both parallel and sequential implementation across all grids from 400by400 to 2000x2000 as specified in the coursework.

Fig. 5.0

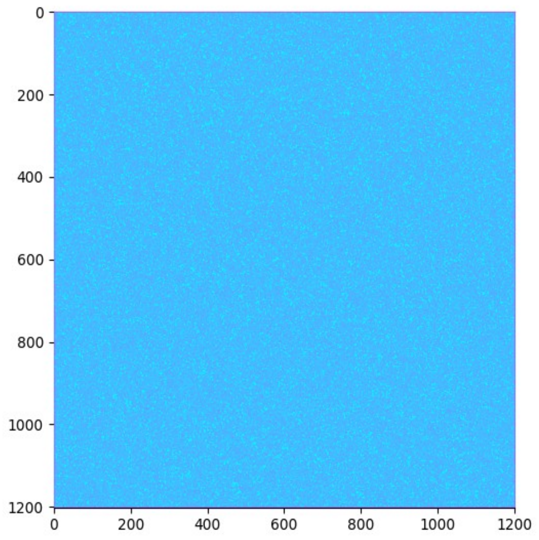
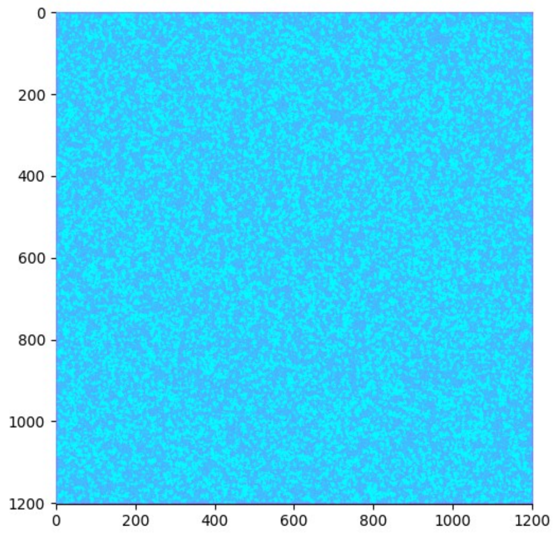
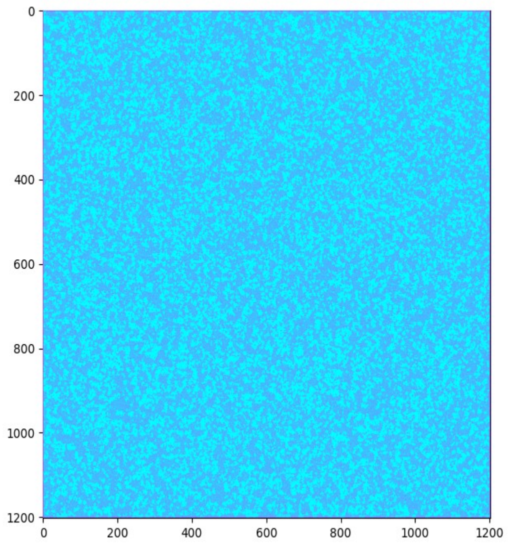


Fig.6.0





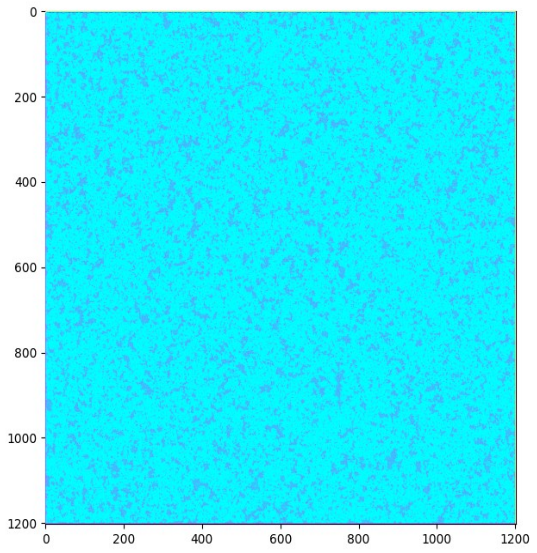


Fig. 7.0 Fig.8.0

RESULTS AND EVALUATION

Simulation would be carried out 4 times using both sequential and parallel processes with the two forms of neighborhood techniques.

Von Neumann (4.0)

GRID SIZE	SEQUENTIAL	PARALLEL
100 x 100	0.306211345690876	3.9516180452728278
400 x 400	5.263746974334717	8.1384779751014709
800 x 800	21.93840012069702	23.086263684463501
1000 x 1000	52.77464008331299	50.77271127700806
1200 x 1200	79.79963397979736	74.67589049847378
2000 x 2000	210.6747838928399	207.7648928837467

Table 1.0

Moore neighborhood(8)

GRID SIZE	SEQUENTIAL	PARALLEL
100 x 100	0.289816395293701	3.431828045272827
400 x 400	8.597136974334717	10.007975101470947
800 x 800	31.18244012069702	33.34563684463501
1000 x 1000	55.23345398902893	53.86026978492737
1200 x 1200	73.51312589645386	71.82765603065491
2000 x 2000	238.23423788999876	234.17683792114258

Table 2.0

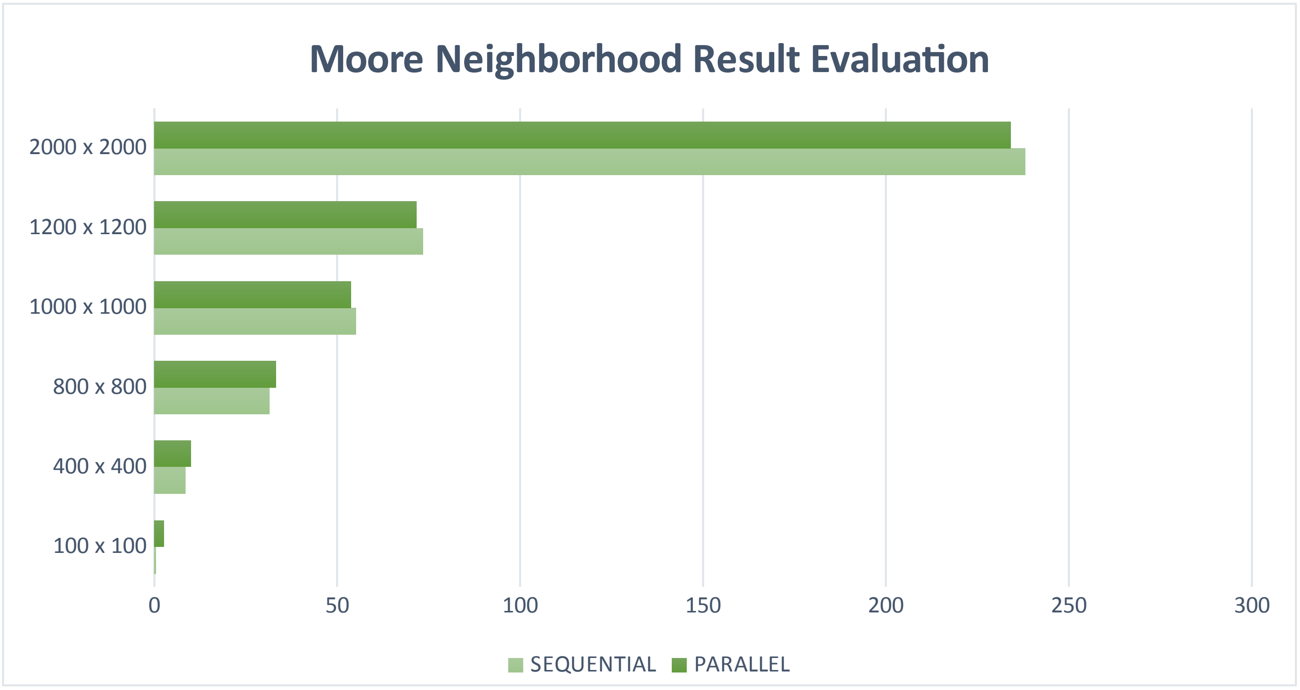


Fig 9.0 Moore Neighborhood Result Evaluation

CONCLUSION

A useful method for simulating the spread of a forest fire is cellular automation. By analyzing how a single site on a grid can change as a result of interacting with other sites on the grid, it simplifies the simulation. When factors like the size of the forest or the neighborhood method are changed, we can also observe the differences in the ways that a fire can spread. Using parallel processing techniques like Numba, we can also observe an improvement in the processing and execution times of our model.Instead of allowing Numba to automate the parallel processes, this model might be strengthened by employing a more reliable parallel processing technique. Additionally, we may add more variables to the simulation to make it more accurate, such as different kinds of trees, wind direction and speed, etc. As said at the outset, the importance of mathematical modelling such as this cannot be overemphasized as it creates a solution to real life problem of controlling wildfires through the in-depth understanding of its behavior, thus limiting the potential damage that could impact human lives.