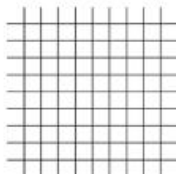


This project was inspired by the Wildfire breakout that happened in California and the Haze crisis that happened annually in Southeast Asia (the worst haze crisis was in 2015).

### 1. Model setup:



We consider an area as a grid with each cell either is of one state:

[1] E: empty

[2] T: or has trees

[3] F: or has trees and those trees are on fire

This area has a certain coverage area, called D (density)  $D = (0, 1]$

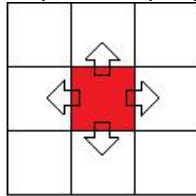
### 2. Model assumptions:

#### a) Step count and Spreading of fire

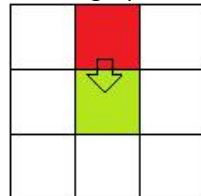
To further simplify this model, I will assume that the fire only spread across row or column, not the diagonal.

-The time is discrete  $t: T = \{t: 0, 1, 2, 3, \dots\}$

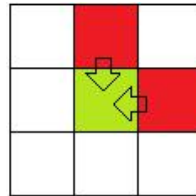
-The probability any square next to a burning square catch fire in the next moment is  $p: P = [0, 1]$



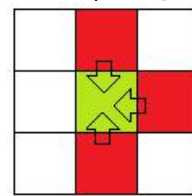
Each neighbour catch fire with probability  $p$



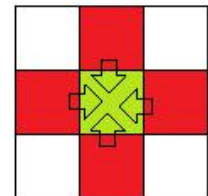
Type 1:



Type 2:



Type 3:



Type 4:

#### b) Initial condition

The causes of fire can be very different, such as lighting (nature disaster) or slash-and-burn practice to clear the area (human activities). The fire can start at multiple spot within the area, can be individual spot at the beginning and merge together and become huge. Multiple fire may start at different point of time. In this report I consider the fire start from one spot only. The time point 0 is when the fire start and it is at center of the region

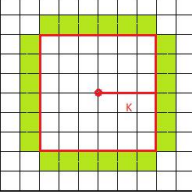
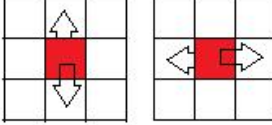
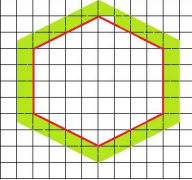
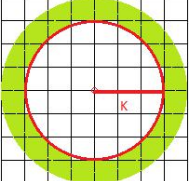
#### c) Fire shapes

Even with all of this simplified assumption, the shape of fire at any point of time is very difficult to predict if it is not one of some special cases ( $p = 1, d = 1$ ) or ( $p = 0$ ). However, since the fire starts at one point and it can spread across 4 different directions, the burnt area can be approximated by a closed geometric shape: square, square diamond, or circle or hexagonal, which has 2 line of symmetry

#### d) Fire speed

Consider a few special cases: Probability of fire:  $P$ , Density:  $D$ ,  $X$  is the total area of the shape at time  $t$ , and the half of diagonal/side of each shape:  $k$ .  $d(\text{Area})$  is the new fire area catching in the next moment

D	P	Burnt area	
0	0		<p>The fire cannot spread in this case</p> $\text{Area}(t) = X$ $\text{Area}(t+1) = X$ $d(\text{Area}) = 0$
1	1		<p>(I am sorry that the shape I draw is a bit misleading)</p> <p>The idea is when <math>D = 1</math> and <math>P = 1</math>, the fire always has this diamond shape as each point opens in both direction. If <math>P \neq 1</math> each cell's probability of burnt next is same as case 2</p> $\text{Area}(t) = X = 2k^2$ $\text{Area}(t+1) = 2(k+1)^2$ $d(\text{Area}) = \text{Area}(t+1) - \text{Area}(t) = 2(k+1)^2 - 2k^2 = 2(2k+1) = 4\sqrt{\frac{X}{2}} + 2 = 2\sqrt{2X} + 2 \quad (1)$
			<p>Note: in fact, if the diamond area is displayed using cells, with each side <math>k</math>, the calculation should be:</p> $\text{Area}(t) = X = k^2 + (k-1)^2$ $\text{Area}(t+1) = (k+1)^2 + k^2$ $d(\text{Area}) = \text{Area}(t+1) - \text{Area}(t) = (k+1)^2 - (k-1)^2 = 4k \quad (2)$ <p>Using (1) and (2) I may approximate the burning area in next moment</p>

			as $d(Area) = 4k = 2\sqrt{2X}$ (3)
1	?		<p>Assume that the area has this square shape, this scenario is same as case 1, each neighbour cell is next to exactly one burning cell</p> $Area(t) = X = k^2$ $Area(t+1) = k^2 + 4k$ $d(Area) = Area(t+1) - Area(t) = 4k = 4\sqrt{X}$
			<p>If I think of the fire spreading direction as 2 dimensions Horizontal and Vertical, at each step, all the fire cells can go horizontal or vertical, this setting can create the pattern above. However, the area burnt in the next moment is <math>2\sqrt{X}</math> instead of <math>4\sqrt{X}</math></p> <p>Also this imply the probability of catching fire in the next step of any adjacent cell is <b>p = 0.5</b> (since there are 2 choices of directions)</p>
1	?		<p>If the area has hexagon shape, with each side is k</p> $Area(t) = X = \frac{3\sqrt{3}}{2} k^2 \quad ; \quad Area(t+1) = \frac{3\sqrt{3}}{2} (k+1)^2$ $d(Area) = Area(t+1) - Area(t) = \frac{3\sqrt{3}}{2} (2k+1) = \frac{3\sqrt{3}}{2} \left( \sqrt{\frac{8X}{3\sqrt{3}}} + 1 \right)$ <p>My simple estimation is: 2 vertical edges cell are type 1, and the other 4 edges, the ratio depend on how far it is from vertical/horizontal line and 45 degree line. E.g. a 30 or 60 degree line has 1/3 of type 1 and 2/3 of type 2 cells because it's 15 degree from 45o line and 30o from horizontal/vertical line</p>
1	?		$Area(t) = X = \pi k^2 \quad ; \quad Area(t+1) = \pi (k+1)^2$ $d(Area) = Area(t+1) - Area(t) = \pi (2k+1) = \left( \sqrt{\pi 4X} + \pi \right)$

So, if I assume  $D = 1$  and the current burnt area is  $X$  cells, the number of cells catch fire in the next step is:

$$p = 0 \quad d(Area) = 0 \quad ; \quad p = 0.5 \quad d(Area) = 2\sqrt{X}$$

$$p = 1 \quad d(Area) = 2\sqrt{2X}$$

There are many other value of  $p$  which I cannot estimate the value of  $d(Area)$ , like the hexagon and circle cases. So, with just the 3 points above, the  $d(Area)$  function I estimate is:  $d(Area) = 2\sqrt{2pX}$   $p \in [0,1]$

Using the same approach above, taken in value of  $D$ , the estimation is

$$d(Area) = 2\sqrt{2DpX} \quad D \in (0,1] \quad p \in [0,1]$$

If at a time  $t$  :  $Area(t) = X_t$

At next point  $t+1$  :  $Area(t+1) = X_{t+1} = X_t + 2\sqrt{2DpX_t}$

Initial condition  $X_0 = 1$

This looks like a non-linear recurrence relationship. But trying to approximate  $X$  as a continuous function of  $t$

$$\text{Let } f(t) = \sqrt{X(t)} \rightarrow X(t) = (f(t))^2$$

$$X_{t+1} - X_t = 2\sqrt{2Dp} f(t) dt \quad ; \quad X'(t) = 2f(t) d(f(t))$$

$$\frac{X_{t+1} - X_t}{dt} = 2\sqrt{2Dp} f(t)$$

$$\frac{2f(t) d(f(t))}{dt} = 2\sqrt{2Dp} f(t)$$

$$\frac{d(f(t))}{dt} = \sqrt{2Dp}$$

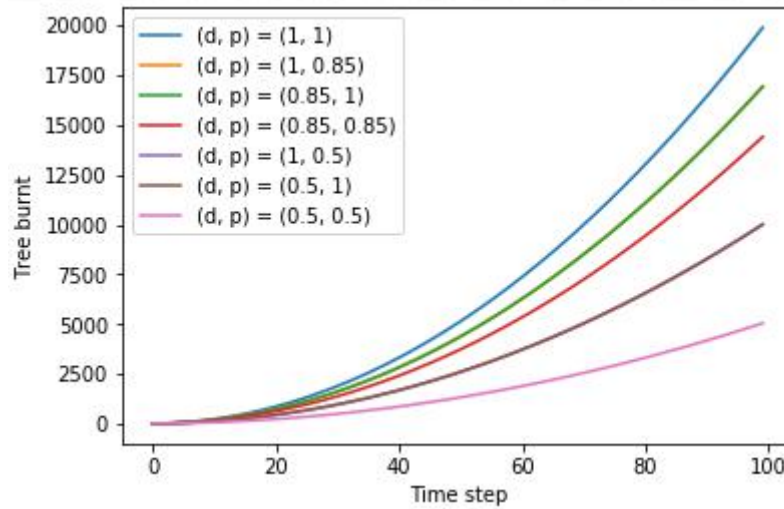
$$f(t) = \sqrt{2Dpt} + C$$

$$X(t) = \left( \sqrt{2Dpt} + C \right)^2$$

Initial condition  $X(0)=1 \rightarrow C=1$

$$X(t) = (\sqrt{2Dpt} + 1)^2$$

Base on this calculation, the rate of burning depend on different Density and Burn probability is:

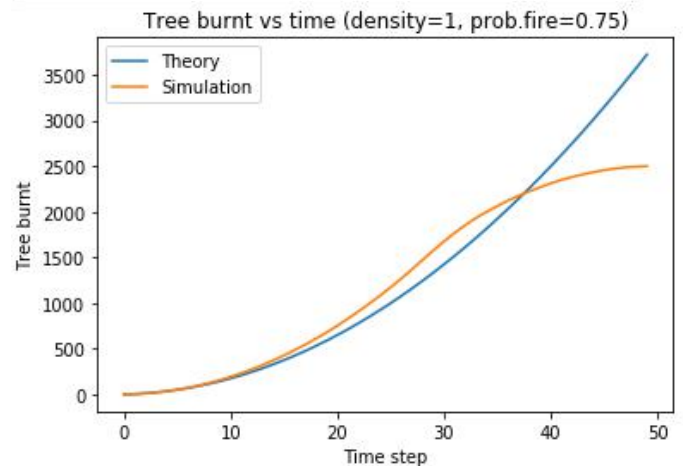
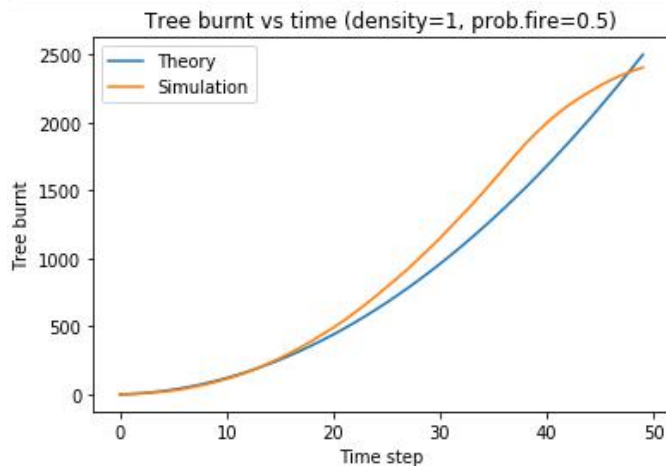
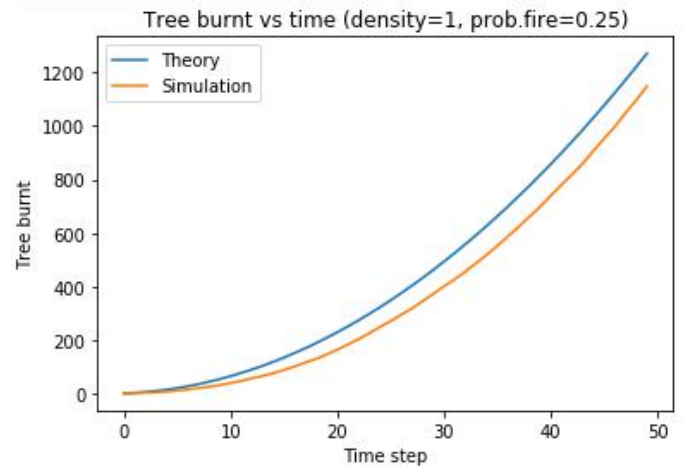
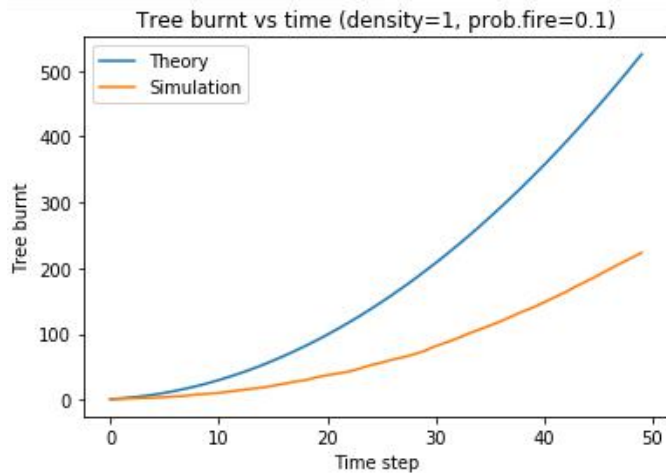


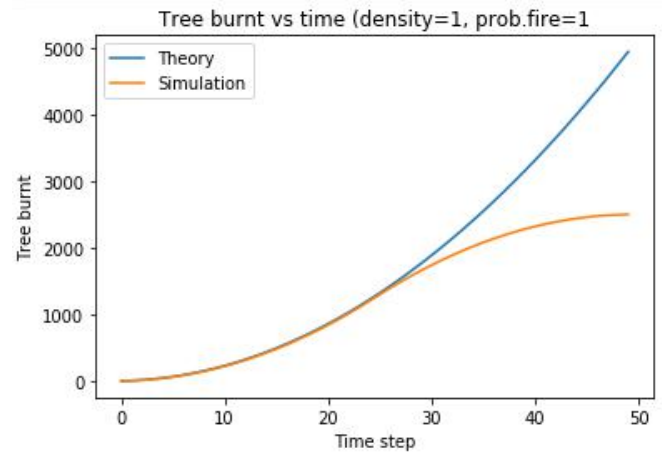
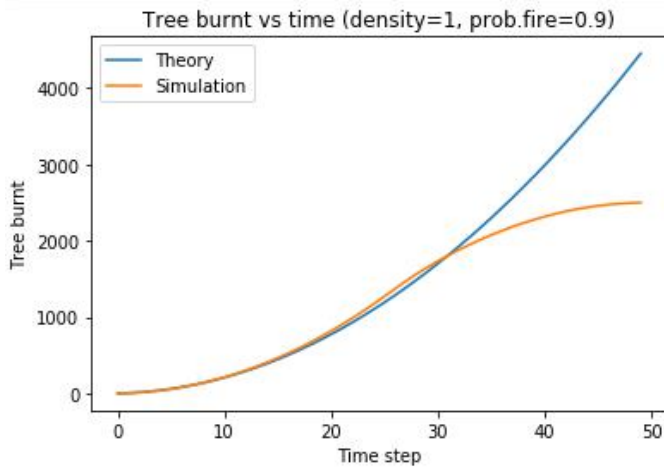
### 3. Simulation result

To simulate, we set up a simple region of 50 x 50 and plot with selected value of Density  $d$  and Catch fire probability  $p$ . The fire starting point is assumed to be the center of the are. The number of replication is 10 (maybe it's not high enough but we are able to observe some behaviour of the model)

#### a) Density = 1

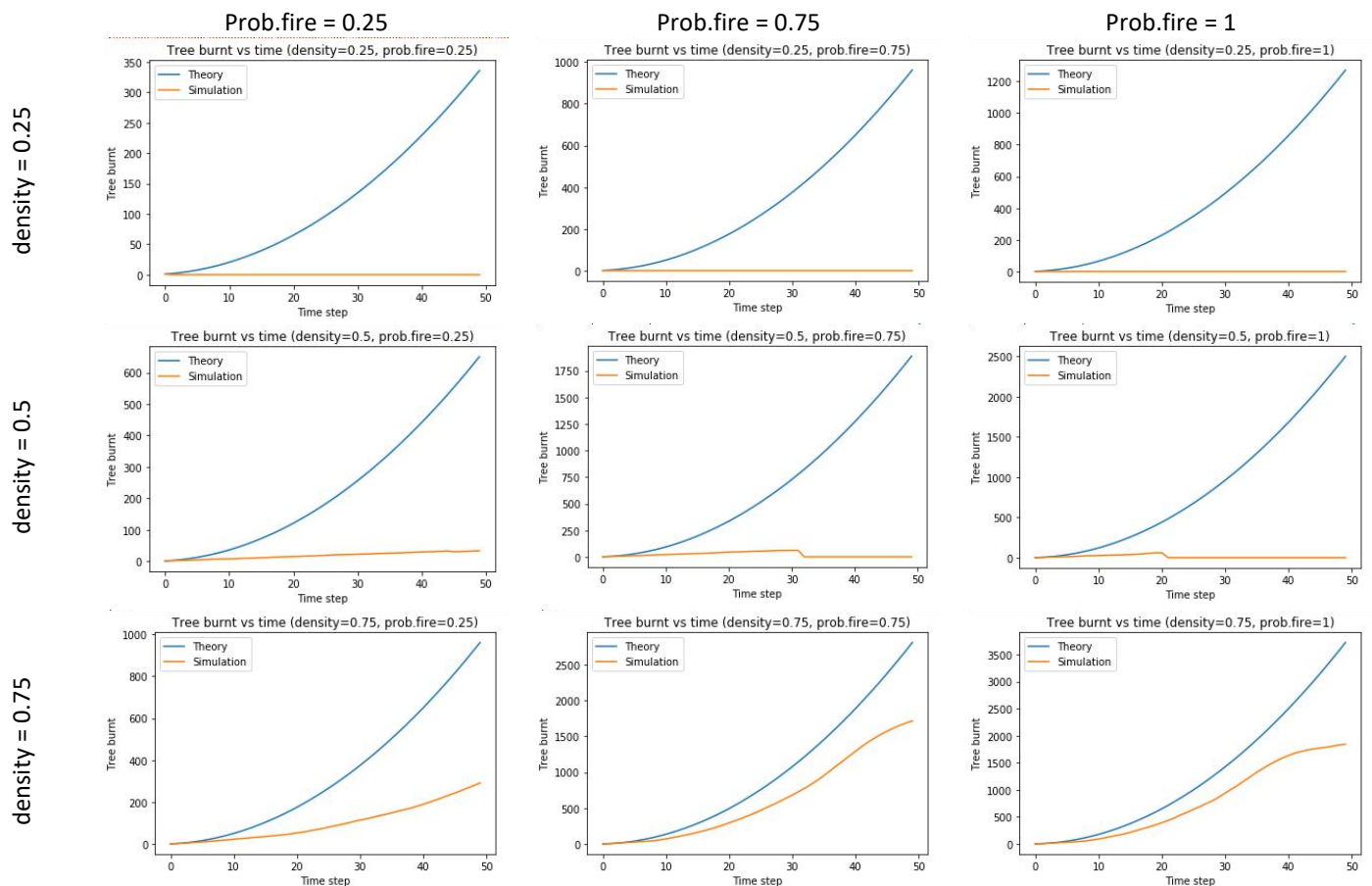
For simple scenario  $d=1$ ,  $p=1$ , the theory and simulation result match. The discrepancy at time step higher than 25 is due to the fact that the fire reach border of the area. At this point, it cannot grow pass the limit of area in simulation which we did not consider in theory. I note that when the probability of fire is very low (0.1) the theory and simulation result is quite different





## b) Density less than 1

The result for lower density of trees is recorded in the table below:



Note: there is a sudden drop in some graph because the number of time taken to burnt is less than simulation time (50 time steps). The reason is the sparse distribution of tree and fire cannot reach the other cluster.

## 4. Conclusion:

- This model approximate well the case of high density and moderate to high probability of fire but fail to works when probability of fire is low or the area is sparse
- This model does not consider the density of tree correctly. I think the reason is I did analysis assuming the fire area is a contained area, which implicitly consider that area has density 1.
- It failed to consider the reduce in tree density after some time step, which is obvious when the area is limited by a boundary
- I considered using Birth-and-death with population dependent to model this but I could not solve the equation yet, and the simulation should be change to accommodate additional parameters.
- I realized after some reading this kind of model is similar to the model of tumor cell growth rather than wildfire behavior.

```

import numpy as np
import random
import matplotlib.pyplot as plt

TREE = 1
EMPTY = 0
ONFIRE = -1

def monitor_forest(x, y, fire_x, fire_y):
    """
    Draw the area map
    """
    plt.scatter(x, y, marker = '.', color = 'g')
    plt.scatter(fire_x, fire_y, marker = '.', color = 'r')
    plt.show()

def reset_forest(forest):
    """
    Reset the forest to initial condition
    """
    for x in xrange(max_x):
        for y in xrange(max_y):
            if forest[x][y] != EMPTY:
                forest[x][y] = TREE

def create_forest(max_x, max_y, tree_density):
    """
    Create the simulation environment
    max_x: int
        length of the area
    max_y: int
        width of the area
    tree_density: float
        density of tree
    """
    x = []
    y = []
    forest = np.zeros((max_x, max_y))
    for j in range(len(forest)):
        for i in range(len(forest[j])):
            if random.random() < tree_density:
                forest[i][j] = TREE
                x.append(i)
                y.append(j)
    return [forest, x, y]

# define neighbour, only in row and column direction
neighbours = [[0, 1],[0, -1],[-1, 0], [1, 0]]

# different value for density and fire probability
prob_fires = [0.25, 0.5, 0.75, 0.9, 1]
tree_densities = [0.25, 0.5, 0.75, 0.9, 1]
# Set parameter of the area and tree density
max_x = 50
max_y = 50
tree_density = 1

for tree_density in tree_densities:
    # Create initial condition
    [forest, x, y] = create_forest(max_x, max_y, tree_density)

    for prob_catch_fire in prob_fires:
        # Number of tree burnt counter

```

```

n_iter = 10
max_time = 50
time_counter = np.zeros(max_time)
tree_counter = np.zeros(max_time)

for i in range(n_iter):
    # Assume fire start at center (just assume a center always has tree)
    reset_forest(forest)
    new_x = [int(max_x / 2)]
    new_y = [int(max_y / 2)]

    # Set up
    fire_x = []
    fire_y = []
    forest[new_x[-1]][new_y[-1]] = ONFIRE
    time_iter = 0
    has_tree = False

    # monitor_forest(x, y, new_x, new_y)
    while (any([len(new_x) > 0, has_tree]) and time_iter < max_time):
        has_tree = False
        fire_x.extend(new_x)
        # print 'fire_x = ' + str(fire_x)
        fire_y.extend(new_y)
        new_x = []
        new_y = []
        time_counter[time_iter] += 1
        tree_counter[time_iter] += len(fire_x)
        time_iter += 1

        for i in xrange(len(fire_x)):
            cur_x = fire_x[i]
            cur_y = fire_y[i]
            for neighbour in neighbours:
                next_x = cur_x + neighbour[0]
                next_y = cur_y + neighbour[1]
                if all([next_x >= 0, next_x < max_x, next_y >= 0, next_y < max_y]):
                    if forest[next_x][next_y] == TREE:
                        has_tree = True
                        if random.random() < probab_catch_fire:
                            forest[next_x][next_y] = ONFIRE
                            new_x.append(next_x)
                            new_y.append(next_y)

coef = (2.0 * tree_density * probab_catch_fire)**(0.5)
t = np.arange(max_time)
x = t * coef + 1
y = x * x
plt.plot(t, y, label = 'Theory')

zidx = np.where(np.equal(time_counter, 0))
time_counter[zidx] = 1
plt.plot(t, tree_counter / time_counter, label = 'Simulation')

plt.xlabel('Time step')
plt.ylabel('Tree burnt')
plt.title('Tree burnt vs time (density=' + str(tree_density) + ', probab.fire=' +
          str(probab_catch_fire)+ ')')

plt.legend()
plt.show()

```