Percolation

Cheng, Jui-Hung

March 3, 2025

1 Introduction

Percolation theory describe the phenomenon when random links or lattice are added to a network or a graph. It is about whether some nodes will be connected, disconnected, or merge into a bigger cluster.

A representative question is as follow. Assume that, there is a material with several holes. Will the liquid be able to reach the bottom if we pour it from the top? The above question can be modelled into the following mathematical question.

Constructing a n*n*n 3-dimensional Vertex Network. The lattices between each two neighbors will be connected with probability p, and unconnected with 1-p and what is the probability p when there is a path that go from the top to the bottom. The above question is now called bond-percolation. There is another question called site-percolation.

1.1 Model explanation

The model I research is a n*n two-dimensional Vertex Network bond-percolation. The lattices between two neighbor Vertex will be connected with probability P that is called bond. Analyze the clusters in the network when the probability is p.

2 Percolation implementation

To implement percolation, I separate the code into five parts.

1. Network Construction
2. Percolation function
3. DFS travel
4. printing cluster information

2.1 Network Construction

To implement the model, we will need a Vertex Network to store the information of the bond. I use a class named Node represent all the possible lattice. That is each Node will have up, down, left right four nodes that can connect to the neighbor nodes and the information that check if the node is connected or not.

```
class Node:
    left  = False
    right = False
    up    = False
    down  = False

    be_connected = False
    count = 0
```

Figure 1: Each node connect to the four direction neighbor nodes

2.2 Percolation function

Traveling the whole map and decide whether connecting each lattice between neighbor nodes with the random function and update the information in both neighbor nodes after adding lattice.

```
def add_lattice(self, x, y, way):
    if( way == "right" ):
        self.nodes[x][y].right = True
    if( way == "down" ):
        self.nodes[x][y].down = True
    if( way == "left" ):
        self.nodes[x][y].left = True
    if( way == "up" ):
        self.nodes[x][y].up = True
    self.nodes[x][y].be_connected = True
```

Figure 2: adding the lattice in one direction but update information in both nodes

However, the lattice in the Vertex Network is bi-directional, that is, we just have to add lattice in down and right direction.

2.3 DFS travel

Using Depth First Search Traveling to count the nodes connected in each cluster. Then DFS traveling the cluster again to label the cluster and add the information to the nodes in the cluster.

```
def travel_the_map(self):
    for i in range(self.L):
        for j in range(self.L):
            if(self.visited[i][j] == False):
                self.dfs_read(i, j)

                if(self.count>1):
                    self.cluster_cur_mark = "[" + chr(self.cluster_no + 65) + "]"

                    if(self.color_text == True):
                        self.cluster_cur_mark = ANSI.color_word(self.cluster_cur_mark)

                    self.cluster_no += 1
                    cluster_item = [self.cluster_cur_mark, self.count]
                    if(cluster_item not in self.cluster_table):
                        self.cluster_table.append(cluster_item)
                else:
                    self.cluster_cur_mark = "[.]"
                    if(self.color_text == True):
                        self.cluster_cur_mark = ANSI.color_word(self.cluster_cur_mark, "grey")

                self.dfs_write(i, j )

                #self.count = 0.0
                self.count = 0
```

Figure 3: DFS traveling two times for counting number in cluster, and adding information in cluster

2.4 Printing cluster information

Collecting the information of cluster when traveling the map, and print the cluster label information, such as : the cluster A connected two nodes, and cluster B connected 20 nodes.

2.5 ANSI color

Using the plain text to print the map is hard to analyzing the cluster distribution, therefore, I import the ANSI_color library to display the map and cluster information with color.

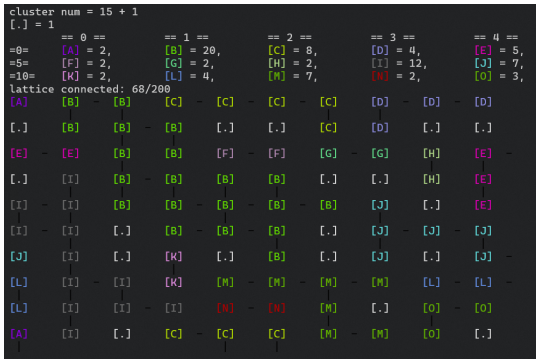

Figure 4: DFS traveling two times for counting number in cluster, and adding information in cluster

3 Analyzing

when the prob is under 0.45, there are at least 25 clusters in a map, and most of the clusters connected at most 50 nodes.

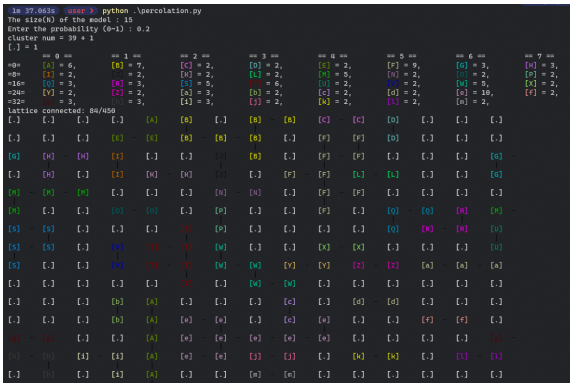

Figure 5: prob = 0.2

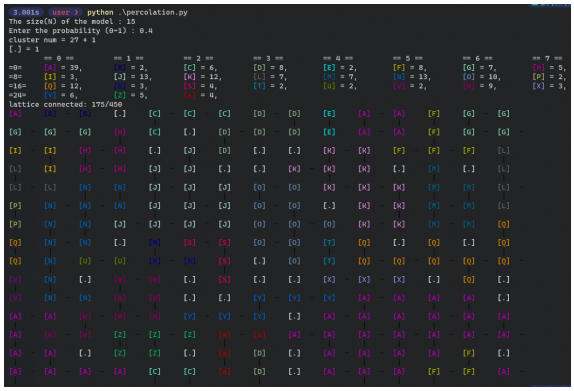

Figure 6: prob = 0.4

when the prob is 0.45, the largest clusters often connected more than 80 nodes,

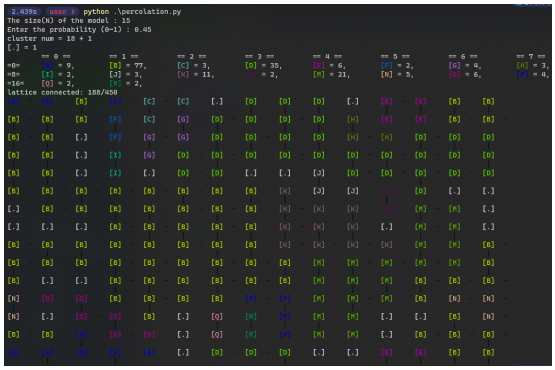

Figure 7: prob = 0.45

when the prob is over 0.45 the number of clusters decrease rapidly when the prob is increase until the 0.6.

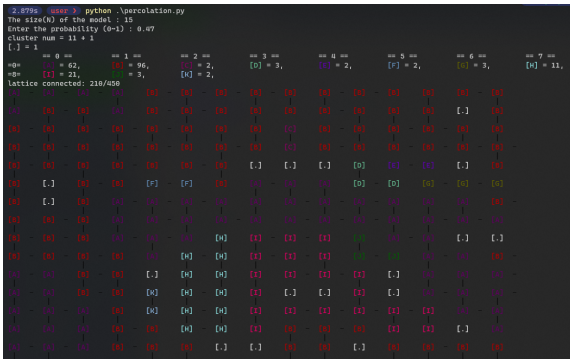

Figure 8: prob = 0.47

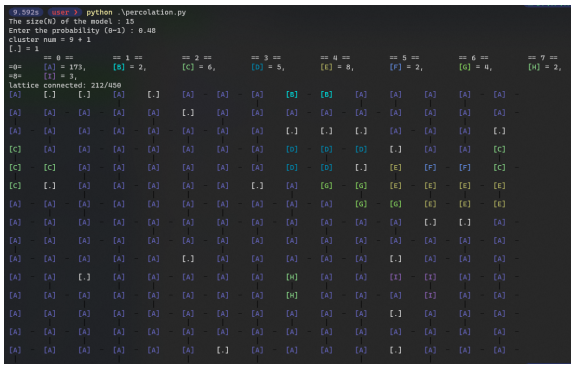

Figure 9: prob = 0.48

when the prob is over 0.55, the number of cluster is usually under 10, and most of the time there

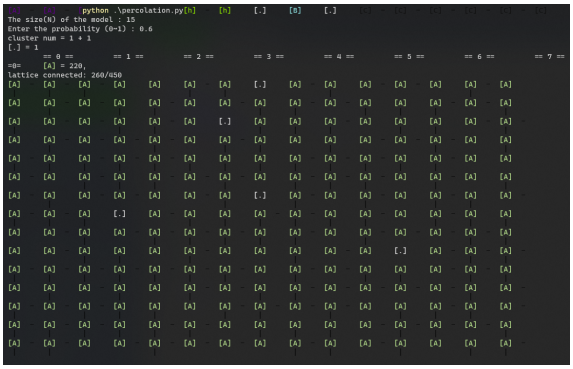

Figure 10: prob = 0.6

Figure 11: prob = 0.7

4 Result

In summary, after testing several time of the percolation the critical value of the probability is 0.45 when the model size is 15. Probability under critical number, there will be many clusters in the map; however, when the probability is over critical value, there will be only one or two large cluster that cover the whole map. The critical value will also increase to 0.5 if the model size is bigger than 15.