1 Introduction

The primary objective of the current exercise is to understand the implementation of the burning algorithm in computing spanning (or infinite) clusters. The sample codes [1] available in `latticeview.zip` folder have been utilized for all tasks, after making specific modifications. The following tasks are sequentially mentioned in this report :

- To draw a square lattice (for studying percolation) with a specfic lattice size ($N \times N$) and occupation probability (p).
- To implement the burning algorithm (to model the forest fires) [3, 1] in finding if a spanning cluster can (or cannot) be achieved. Additionally, computations for the shortest path (in time steps) and total life time of the fire has also been implemented.
- The final task attempts to calculate the statistics for the probability of finding a spanning cluster, by plotting the graphs of wrapping probability (or fraction of percolative cluster), shortest path and total life time of fire versus p. The lattice size (N) and p have been varied for this, in order to capture the plot accurately, whose results are subsequently compared with the literature. The threshold occupation probability (p_c) at which the system changes its behavior has also been obtained from these graphs.

The relevant codes for each task are named as `task1.cpp, task2.cpp` and `task3.cpp` respectively.

2 Algorithmic Description

For all the tasks, a high quality random generator available in C/C++ : `drand48()` defined in the C++ library `stdlib` has been used. The color specifications for each lattice cell (available in `latticeview.h`) are listed in the Table 1.

Table 1: Color specifications for each lattice cell

Color	**Physical meaning of the color**	**Value of the `lat` array element**
White	Empty lattice cell	0
Green	Occupied lattice cell	1
Red	Burning lattice cell	2
Black	Burned out lattice cell	3

2.1 Task 1

For a fixed p in this task, random numbers (varying from 0 to 1) have been produced for each lattice cell. On comparing this random number (p_{cell} as defined in the code) with p, two cases arise :

- $p_{cell} < p$ - In this case, the cell is assumed to be occupied. The `lat` array for such lattice elements in the code is thus initialized to 1 (corresponding to the green color as shown in Table 1).
- If the above condition fails, the cell is considered to be empty, by initializing the corresponding `lat` array elements to 0 (this corresponds to white color).

The code snippet illustrates the above steps as follows :

[1] `http://www.ifb.ethz.ch/education/bsc-courses/bsc-intro-comphys.html`

```
for(int i=0; i < N; i++)
        for(int j=0; j < N; j ++)
        {
                p_cell = drand48();
                if (p_cell < p)
                lat[i+j*N] = 1; //Empty cells - white
                else
                lat[i+j*N] = 0; //Occupied cells - green
        }
```

Note that the two-dimensional array corresponding to each lattice position is converted to a one-dimensioal array `lat`, which corresponds to a position `i+j*N`, where `i,j` are the lattice indices.

2.2 Task 2

The second task deals with the implementation of an algorithm in fighting forest fires (from a figurative sense). Practically, this finds application in understanding the concept of percolation by assuming each tree to resemble a lattice cell. More detailed theoretical information on this topic can be seen in [1]. Initial portions of this task in coloring the occupied sites to green follow from Task 1. The remaining portions of the code follow the burning algorithm, also outlined in [3], as follows :

1. At time step (t) equal to 1, all occupied cells will be colored green, and the remaining ones will be white (Task 1). Label these occupied cells as 1.

2. At the next time step ($t = 2$), all the occupied lattice cells (green) in the first row will be burned (figuratively). In other words, `lat` corresponding to their positions will be marked red. Label the burning lattice cells with the corresponding time step value (implemented by `label` array in the code).

3. For each time step ($t + 1$), check the lattice cells whose label corresponds to t (burning). Label their immediate occupied neighbors (whose label is 1) with an updated label $t + 1$. `lat` at the lattice cell positions should be initialized to 2 also, as these have just started burning. Additionally, `lat` at the lattice cells with label t should be equated to 3 (burned out - black color).

4. The step 3 should be repeated until there are no other neighbors to be burned.

A spanning cluster is said to be found if the bottom line of the lattice has been reached (so as to infinitely percolate). Once the bottom line has been reached at the time instant $t + 1$, the shortest path will correspond to a number of time steps equal to t. Additionally, the step 3 in the above algorithm has to be repeated until all the cells are burned out, in order to capture the total life time of fire. The image files of the lattice at each time instant are then converted into a GIF image, to visualize the animation.

Figure 1 beautifully illustrates the conditions when a spanning cluster can be achieved or not.

2.3 Task 3

The code in the Task 2 is extended to accomplish this task. To capture the statistics of the percolation probability, the number of seed measurements are stored in the variable `no_seeds`. The code in Task 2 is thus repeated `no_seeds` times, so as to compute the average shortest path (in case of a spanning cluster), average total life time of the fire and fraction of percolative clusters. For this, the seeding of the `drand48()` function in the code is performed by a seeding function - `srand48()`. Later, the curves are plotted for various values of N and p, as detailed in the Discussion section.

3 Results

3.1 Task 1

The square lattice was drawn from the C++ header file - `latticeview.h`. This has been captured for two cases of p, as shown in Figures 2 and 3.

3.2 Task 2

For specific values of N and p, certain images of the GIF animation have been captured, as shown in Figures 4 - 7. For each Figure, the shortest path (if the spanning cluster has been acheieved) and the total life time of the fire in time steps have also been mentioned.

3.3 Task 3

The graphs of fraction of percolative cluster, average shortest path and average time of the fire are plotted in Figure 8, for various lattice sizes, averaged over 100 measurements. Additionally, the threshold occupation probability p_c is also suitably captured in each Figure.

4 Discussion

4.1 Task 1

Results for the Task 1 clearly show that the second lattice (for the case $p = 0.7$) is more densely packed, as it has a high probability of occupying lattice cells. Thus, one would expect all empty lattice cells for $p = 0$ and all occupied lattice cells at $p = 1$.

4.2 Task 2

Results of this task show that higher the occupation probability for a fixed N, higher are the chances of finding a percolative cluster (as the lattice will be more densely packed - observation from Task 1). On similar lines, if the lattice is densely packed, the shortest path (in case of spanning cluster) and life time of fire will require less number of time steps to be reached.

4.3 Task 3

From Figure 8 pertaining to wrapping probability, we can observe that as the grid size of the lattice increases (from 100×100 to 1000×1000), the curves tend to the threshold probability p_c. For Average shortest path, the grid sizes 500×500 and 1000×1000 seem to exhibit a better prediction than the coarse size of 100×100. Once again, the peak average shortest path (for the spanning cluster) is noticed at p_c. Additionally, the average shortest path assumes relatively low values (close to 0) below p_c (not observed in 100×100 due to the coarseness of the lattice size and also due to averaging over a lesser number of measurements). For the case of Average total fire time, similar peaks are noticed near p_c. However, the total fire time in this case retains values sufficiently greater than 0 (unlike the case of shortest path), as this does not require the condition of spanning cluster to be achieved. The credibility of the obtained results is confirmed on comparison with previous results in the literature [2, 3, 1].

References

[1] Amnon Aharony and Dietrich Stauffer. *Introduction to percolation theory.* Taylor & Francis, 2003.

[2] Tom Beer and I.G. Enting. Fire spread and percolation modelling. *Mathematical and Computer Modelling*, 13(11):77 – 96, 1990.

[3] Hans Hermann. Lecture notes on introduction to computational physics. Institute for Building Materials, ETH Zürich, 2017.

[4] Robert Sedgewick and Kevin Wayne. Case study : Percolation.

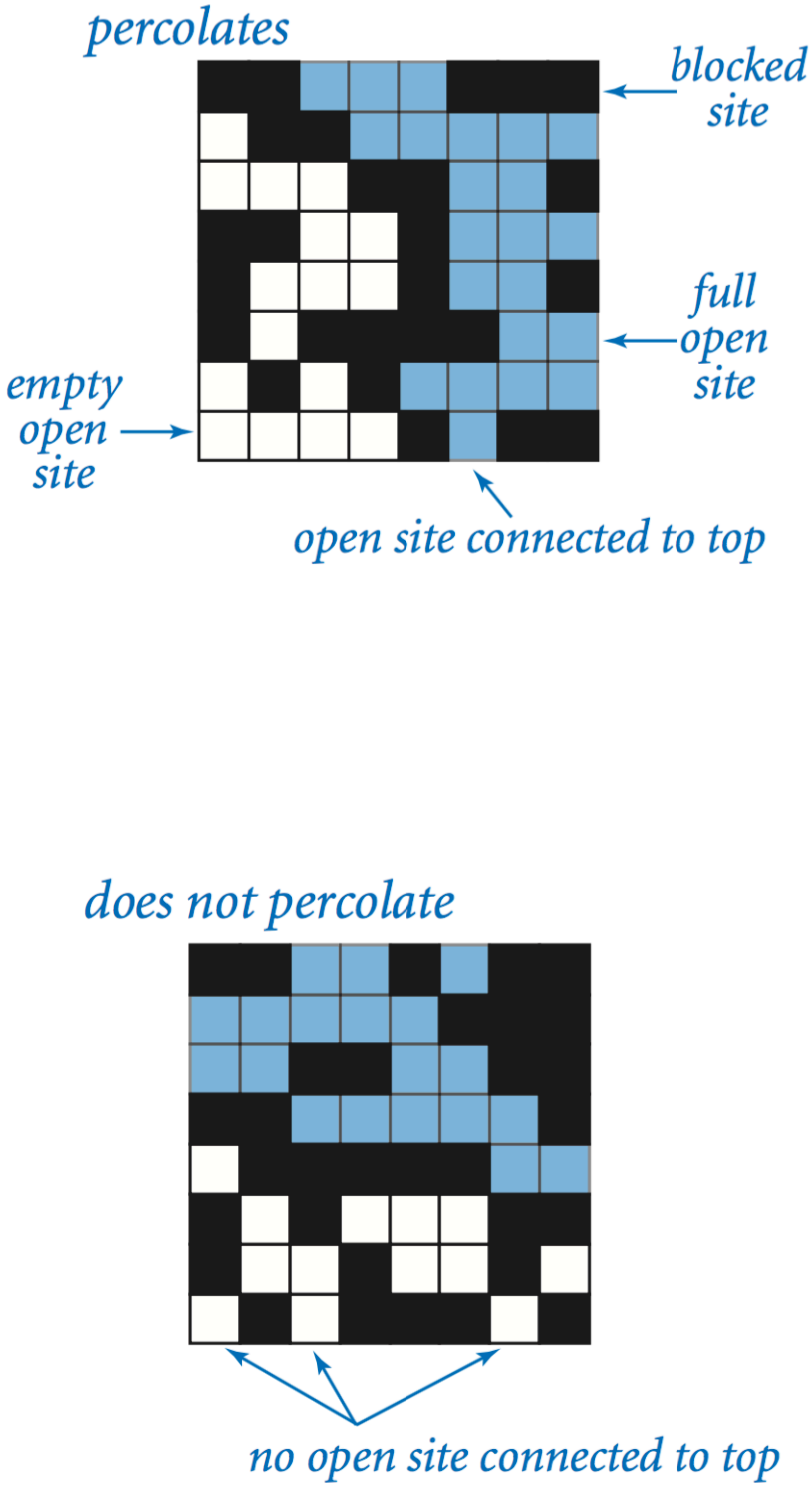


Figure 1: Illustration to explain the possibility of a spanning cluster [4]

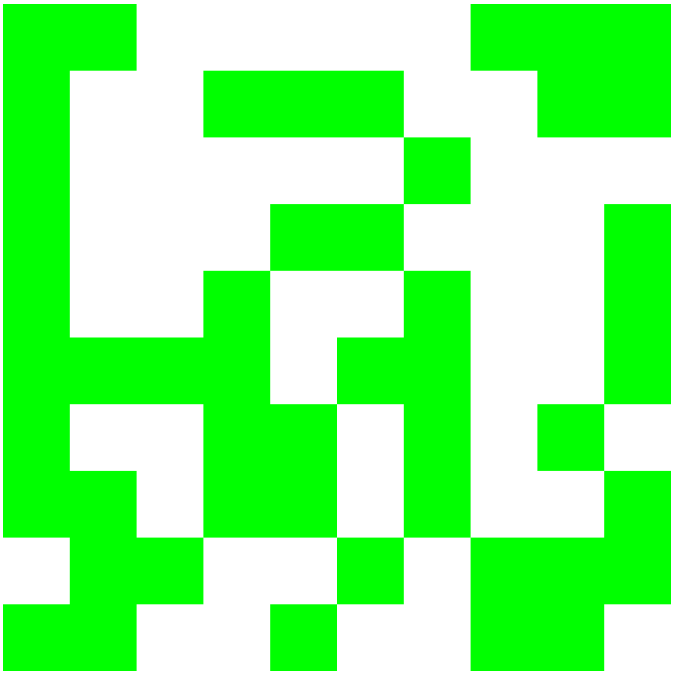


Figure 2: Visualization of the lattice captured from the code at $p = 0.5$

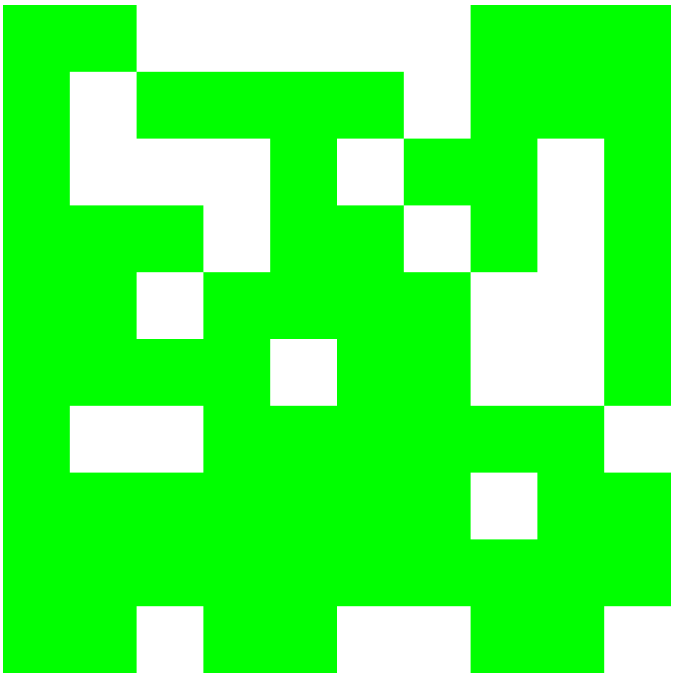
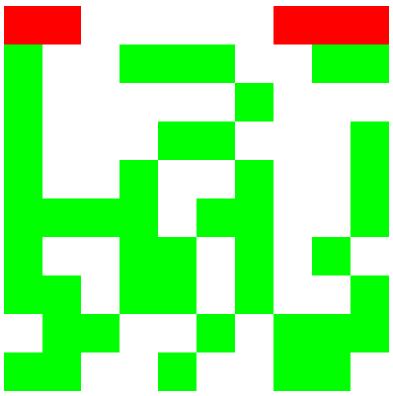
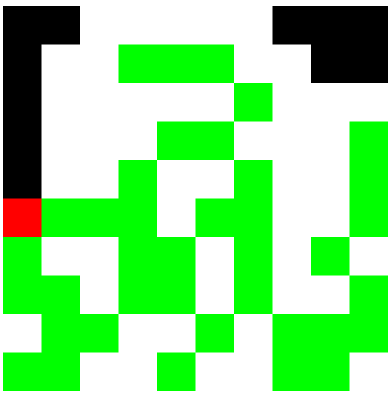


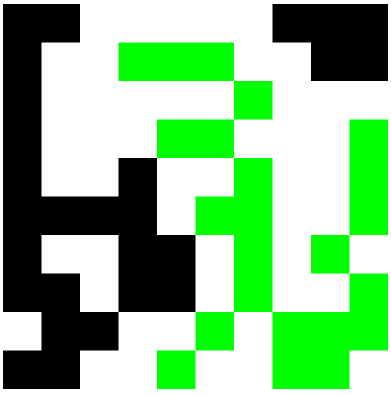
Figure 3: Visualization of the lattice captured from the code at $p = 0.7$



(a) $t = 2$

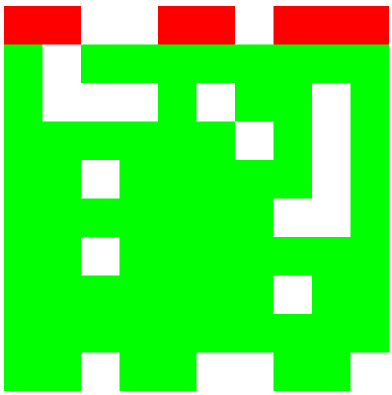


(b) $t = 7$

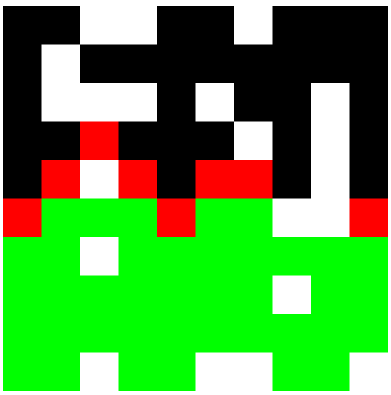


(c) $t = 14$

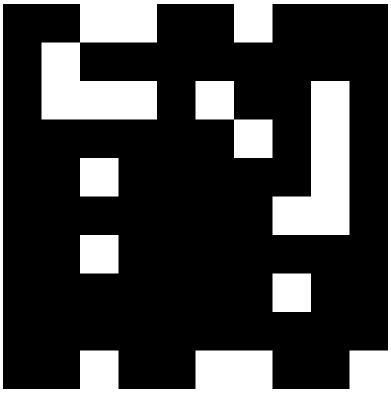
Figure 4: Lattice images captured at $N = 10$ and $p = 0.5$, Shortest path = 13 and Toal life time of fire = 14 (in time steps)



(a) $t = 2$

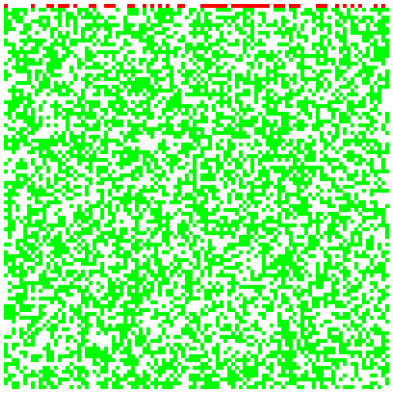


(b) $t = 7$

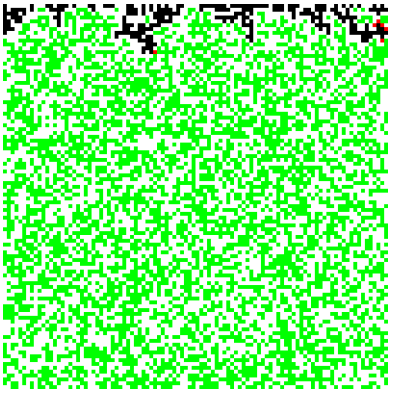


(c) $t = 14$

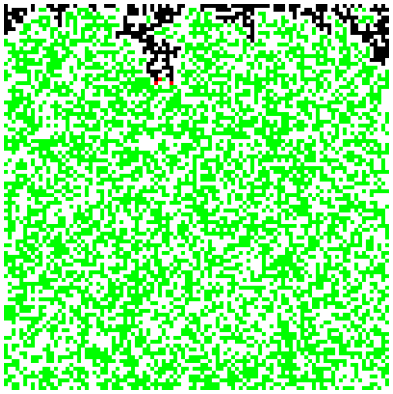
Figure 5: Lattice images captured at $N = 10$ and $p = 0.8$, Shortest path $= 11$ and Toal life time of fire $= 14$ (in time steps)



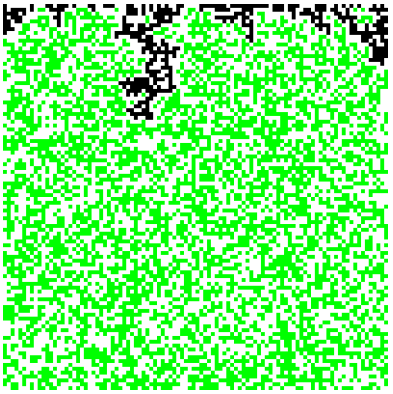
(a) $t = 2$



(b) $t = 23$

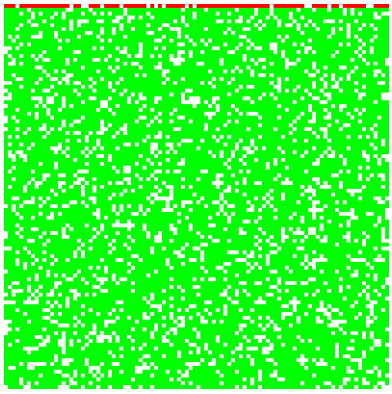


(c) $t = 35$

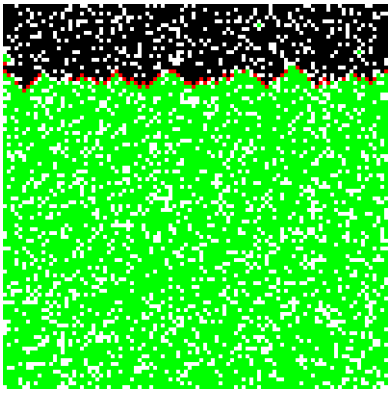


(d) $t = 56$

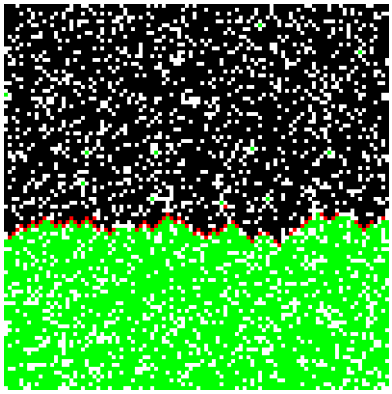
Figure 6: Lattice images captured at $N = 100$ and $p = 0.5$, No spanning cluster can be achieved for this configuration, Toal life time of fire = 56 (in time steps)



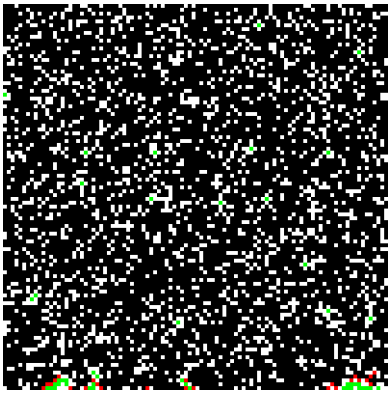
(a) $t = 2$



(b) $t = 24$

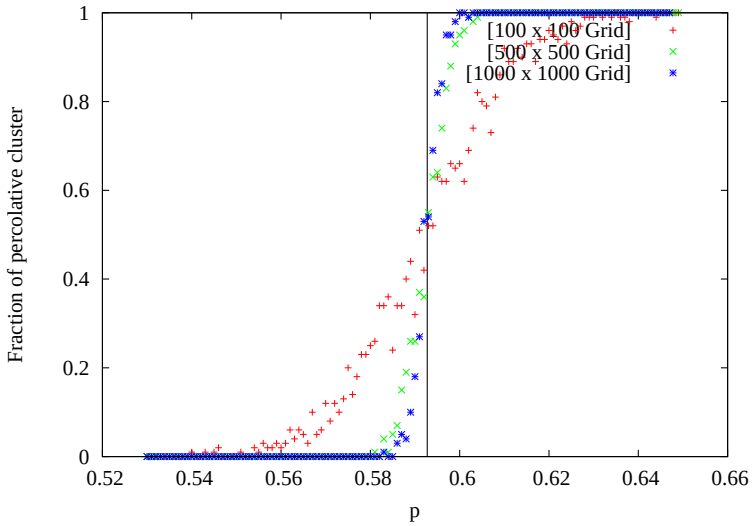


(c) $t = 68$

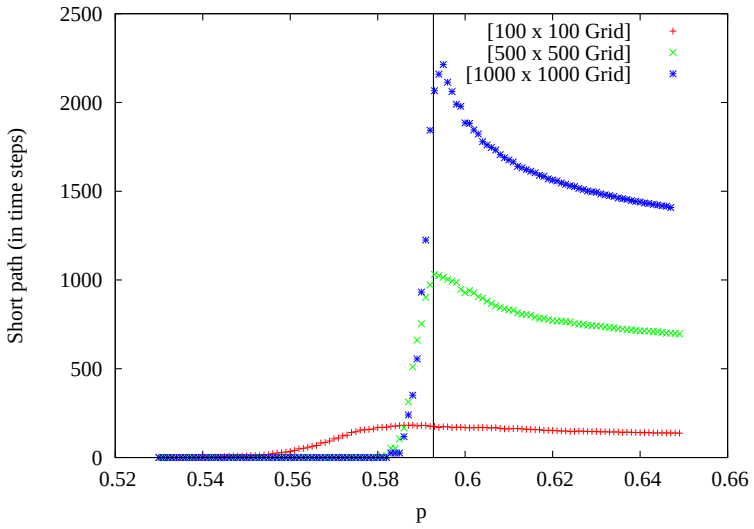


(d) $t = 118$

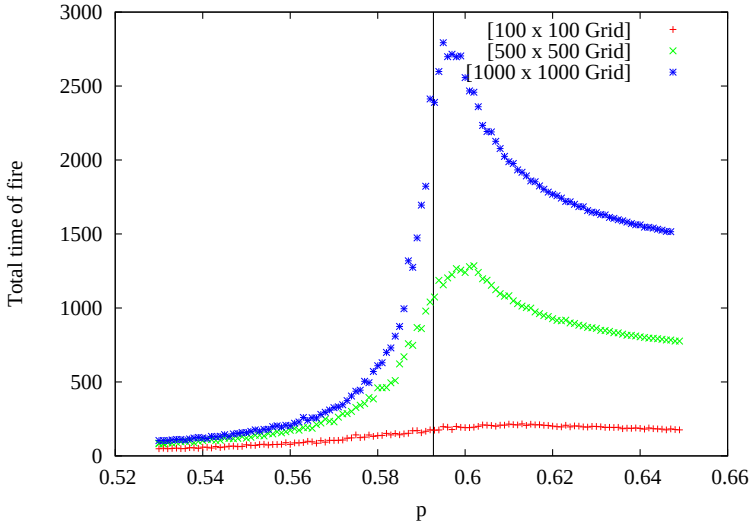
Figure 7: Lattice images captured at $N = 100$ and $p = 0.8$, Shortest path = 109 and Toal life time of fire = 123 (in time steps)



(a) Fraction of percolative cluster (Wrapping probability)



(b) Average shortest path



(c) Average total fire time

Figure 8: Graphs of wrapping probability, Average shortest path and Average total fire time vs occupation probability