Computational Physics Project

Many Body Physics - Week 1: Percolation Theory

Submitted by
Ramit Goyal
Roll No.: 25B1049

Mentor
Arnav Jain

Maths and Physics Club, IIT Bombay

December 24, 2025

Contents

1 Introduction

Percolation theory provides a probabilistic model for studying connected clusters in a random graph. It is fundamental to understanding phase transitions, critical phenomena, and transport properties in disordered systems.

In this project, we implement a numerical simulation of **Site Percolation** on a 2D square lattice of size $L \times L$. Sites are randomly occupied (filled black) with probability p. The primary objective is to investigate the phase transition that occurs at a critical probability p_c, where a cluster emerges, connecting opposite boundaries of the lattice.

2 Computational Methods

The simulation is built upon four primary components: Grid Initialization, Cluster Identification (Hoshen-Kopelman), Percolation Detection, and Monte Carlo Analysis.

2.1 1. Grid Initialization

We generate an $L \times L$ grid where each site (i, j) is occupied (1) with probability p and empty (0) with probability $1 - p$. This is achieved by generating a matrix of uniform random numbers $r \in [0, 1)$ and applying a threshold.

Algorithm 1 Grid Initialization

```
function GenerateGrid(n, p)
    R ← random n × n matrix with values in [0, 1)
    grid ← (R < p)                          ▷ Boolean matrix: True if occupied
    return grid
end function
```

2.2 2. Cluster Identification: Hoshen-Kopelman Algorithm

To identify distinct clusters, we implement the **Hoshen-Kopelman algorithm** enhanced with a **Union-Find** data structure. This algorithm scans the grid once to assign provisional labels and resolve equivalences between connected neighbors.

Connectivity Rule: Sites are connected if they are nearest neighbors (Up, Down, Left, Right). In the first pass (Top-to-Bottom, Left-to-Right), we only need to check the Left and Up neighbors.

Algorithm 2 Hoshen-Kopelman with Union-Find (User Implementation)

```
function HoshenKopelman(grid)
    n ← length of grid
    labels ← n × n zeros matrix
    label_counter ← 0
    parent ← dictionary for Union-Find
    Helper: Find(x) returns root of x (with path compression)
    Helper: Union(x, y) links root of y to root of x
    for i ← 0 to n − 1 do
        for j ← 0 to n − 1 do
            if grid[i][j] is Occupied then
                neighbors ← []
                if i > 0 and labels[i − 1][j] > 0 then neighbors.append(labels[i − 1][j])
                end if
                if j > 0 and labels[i][j − 1] > 0 then neighbors.append(labels[i][j − 1])
                end if
                if neighbors is empty then
                    label_counter ← label_counter + 1
                    labels[i][j] ← label_counter
                    parent[label_counter] ← label_counter
                else
                    min_label ← min(neighbors)
                    labels[i][j] ← min_label
                    for neighbor in neighbors do
                        Union(min_label, neighbor)
                    end for
                end if
            end if
        end for
    end for
                                                    ▷ Second Pass: Flatten labels
    for i ← 0 to n − 1 do
        for j ← 0 to n − 1 do
            if labels[i][j] > 0 then
                labels[i][j] ← Find(labels[i][j])
            end if
        end for
    end for
    return labels
end function
```

2.3 3. Percolation Detection

We define the system as percolating if there is a cluster label present in the top row that is also present in the bottom row.

Algorithm 3 Check Percolation

```
function CheckPercolation(labels)
    n ← length of labels
    top_set ← unique labels in row 0 (excluding 0)
    bottom_set ← unique labels in row n − 1 (excluding 0)
    common ← top_set ∩ bottom_set
    if common is not empty then
        return True
    else
        return False
    end if
end function
```

3 Simulation Parameters & Analysis

We performed Monte Carlo simulations to calculate statistical averages.

- **Lattice Sizes (L):** 8, 16, 32.
- **Probability (p):** Linearly spaced values from 0 to 1.
- **Trials:** Averaged over N realizations per (L, p) pair (e.g., 100-300 trials).

3.1 Calculated Quantities

3.1.1 1. Weighted Average Cluster Size (S)

This measures the expected size of a cluster chosen by picking a random site.

$$S(p) = \frac{\sum_s s^2 n_s}{\sum_s s n_s} \tag{1}$$

where n_s is the number of clusters of size s.

3.1.2 2. Percolation Strength (P)

The order parameter of the phase transition. It is the fraction of sites belonging to the percolating cluster.

$$P = \frac{\text{Size of percolating cluster}}{L^2} \tag{2}$$

If no percolation occurs, P = 0.

3.1.3 3. Percolation Rate / Probability (Π)

The probability that a random lattice at occupation p contains a spanning cluster.

$$\Pi(p) = \frac{\text{Count of Percolating Trials}}{\text{Total Trials}} \tag{3}$$

4 Results and Observations

4.1 Visualizing the Lattice

Before analyzing the statistics, we visualized the grid to understand the qualitative behavior of clusters. Figure 1 shows the lattice state for a specific configuration.

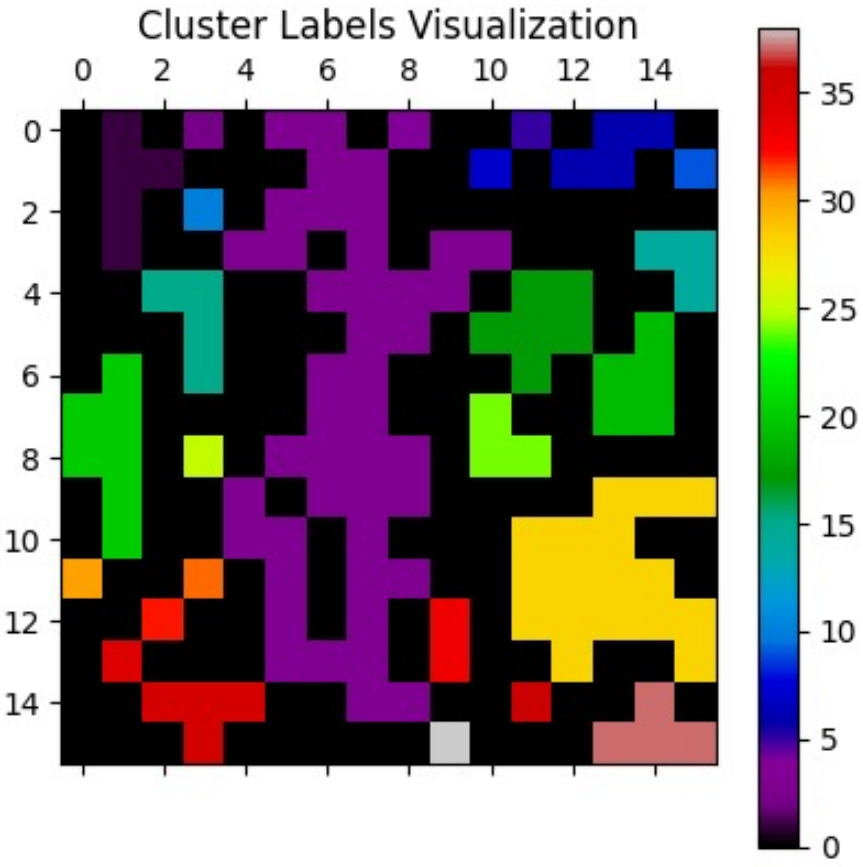

Figure 1: Visualization of clusters on a 2D square lattice for **n=16** and **p=0.5**. Distinct colors represent different connected components identified by the Hoshen-Kopelman algorithm.

4.2 Phase Transition Analysis: Site Percolation

We analyzed the system behavior for lattice sizes $L \in \{8, 16, 32\}$. The following plots illustrate the phase transition behavior.

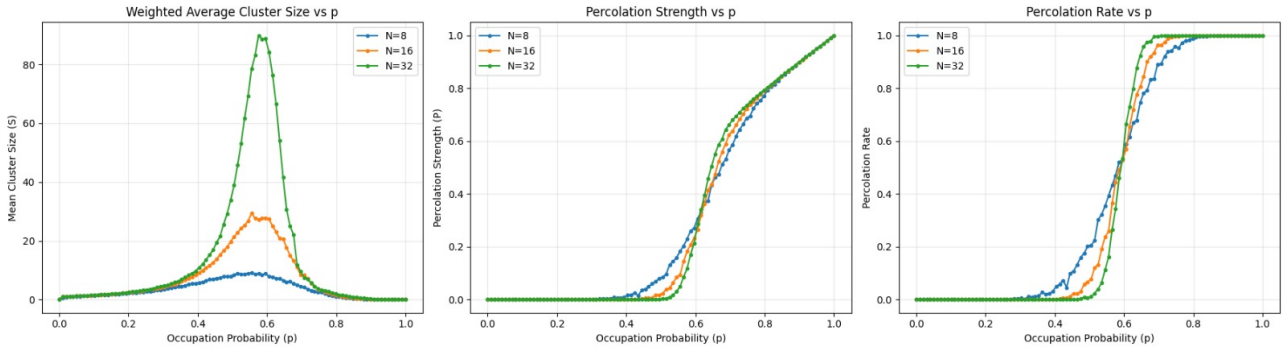

Figure 2: Site Percolation Results: (Left) Percolation Strength P_∞, (Center) Percolation Probability $\Pi(p)$, and (Right) Average Cluster Size S vs. Probability p.

From Figure 2, we observe:

4.2.1 Percolation Probability

- Exhibits a sigmoid (S-shape) transition from 0 to 1.
- As L increases, the transition becomes sharper (step-function like).
- All curves intersect near **$p \approx 0.59$**, identifying the critical probability for site percolation.

4.2.2 Percolation Strength (P_∞)

- For $p < p_c$, the strength is effectively zero.
- At $p > p_c$, there is a sharp increase as the giant component forms.
- This confirms P_∞ as the order parameter for the transition.

4.2.3 Weighted Average Cluster Size (S)

- Shows a distinct peak near $p_c \approx 0.59$.
- **Finite Size Scaling:** The height of the peak increases dramatically with system size L, indicating the divergence of correlation length at the critical point.

4.3 Comparison with Bond Percolation

In addition to site percolation, we simulated **Bond Percolation** on the same square lattice. In this model, edges (bonds) between sites are occupied with probability p, rather than the sites themselves.

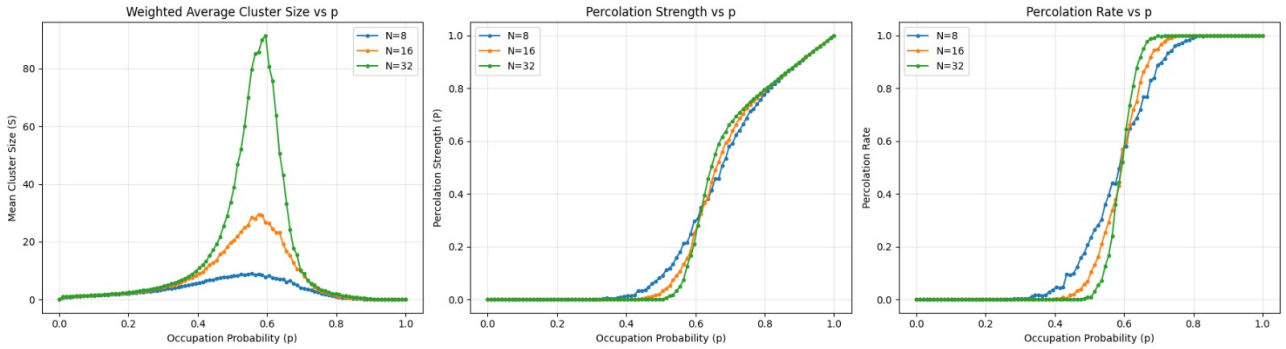

Figure 3: Bond Percolation Results for $L = 8, 16, 32$. The transition occurs distinctly earlier than in site percolation.

As seen in Figure 3, the qualitative behavior of the phase transition remains the same (sigmoid probability, diverging average size), but the critical threshold shifts:

- Site Percolation p_c: ≈ 0.5927
- Bond Percolation p_c: ≈ 0.5000

This aligns with the exact theoretical result for bond percolation on a 2D square lattice, where $p_c = 1/2$.

4.4 Theoretical Comparison: The Bethe Lattice

It is instructive to compare our 2D lattice results with the Bethe Lattice, which represents an infinite-dimensional system where mean-field theory is exact. For percolation to occur, a branch must, on average, produce at least one continuing branch. Since one bond enters a site, there are z 1 outgoing paths. For a Bethe lattice with coordination number z (number of neighbors), the critical probability is given by:

$$p_c = \frac{1}{z-1} \tag{4}$$

5 Conclusion

Our simulation successfully replicates the critical behavior of percolation on a 2D square lattice. We observed:

1. Site Percolation: A phase transition at $p_c \approx 0.59$.
2. Bond Percolation: A phase transition at $p_c \approx 0.50$.
3. Finite Size Effects: Larger systems ($L = 32$) exhibit sharper transitions and higher susceptibility peaks than smaller systems ($L = 8$).