Υπολογιστική Φυσική Στερεάς Κατάστασης
Percolation

Σεϊτανίδου Δήμητρα

14 Ιουλίου 2020

Το πρόβλημα της διήθησης

Στην παρούσα εργασία μελετάμε το φαινόμενο της διήθησης φτιάχνοντας ένα δισδιάστατο πλέγμα το οποίο έχει δύο μόνο ειδών θέσεις, ανοιχτές ή κλειστές, που συμβολίζονται με 0 ή 1 αντίστοιχα. Η αναλογία των 0 και 1 ορίζεται από την πιθανότητα p. Οι κλειστές θέσεις δημιουργούν συμπλέγματα (clusters) όταν δίπλα τους υπάρχουν άλλες κλειστές θέσεις (πάνω, κάτω, δεξιά και αριστερά). Σκοπός μας είναι να μετρήσουμε αυτά τα συμπλέγματα και να βρούμε το μέγεθος τους. Για να το κάνουμε αυτό χρησιμοποιούμε έναν αλγόριθμο που ονομάζεται Cluster-Multiple-Labelling-Technique (CMLT).

CMLT

Σκοπός του αλγορίθμου CMLT είναι να βρούμε το μέγεθος των clusters και να τα ονομάζουμε μόνο με ένα πέρασμα από το δισδιάστατο πλέγμα. Κοιτάμε τα στοιχεία ένα-ένα κατά στήλη και τα χωρίζουμε σε clusters ανάλογα αν υπάρχει κλειστή θέση επάνω ή αριστερά από την εκάστοτε κλειστή θέση που κοιτάμε. Για κάθε ένα cluster κρατάμε το όνομα του (label), που ξεκινάει από 1 και αυξάνεται για κάθε καινούργιο cluster και το μέγεθος του. Όταν δύο clusters ενώνονται, το όνομα του συνενωμένου cluster γίνεται, κατά προτεραιότητα, πρώτα αυτό του αριστερού και αν δεν υπάρχει του επάνω. Επίσης ενώνονται και τα μεγέθη τους.

Η άσκηση

Οι ποσότητες που πρέπει να υπολογίσουμε είναι οι:

$$I_{av} = \sum_{m=1}^{m_{max}} \frac{i_m m^2}{pN^2} \tag{1}$$

$$I'_{av} = \sum_{m=1}^{m_{max}-1} \frac{i_m m^2}{pN^2} \tag{2}$$

$$P_{max} = \frac{m_{max}}{pN^2} \tag{3}$$

όπου i_m το πλήθος των cluster με μέγεθος m και το γινόμενο pN^2 το πλήθος των κλειστών θέσεων στο cluster. Με m_{max} συμβολίζεται το μέγεθος του μεγαλύτερου cluster.

Αυτές τις σχέσεις τις υπολογίζουμε για διάφορες τιμές της πιθανότητας p (από 0.1 μέχρι 0.8) και φτιάχνουμε τα διαγράμματα $I_{av}-p$, $I'_{av}-p$, $P_{max}-p$. Από τα διαγράμματα βρίσκουμε την κρίσιμη πιθανότητα p_c.

Έχοντας βρει την κρίσιμη πιθανότητα μπορούμε να υπολογίσουμε τους κρίσιμους εκθέτες β, γ, γ' σύμφωνα με τις σχέσεις:

$$I_{av}(p) = k|p - p_c|^{-\gamma}, p < p_c \tag{4}$$

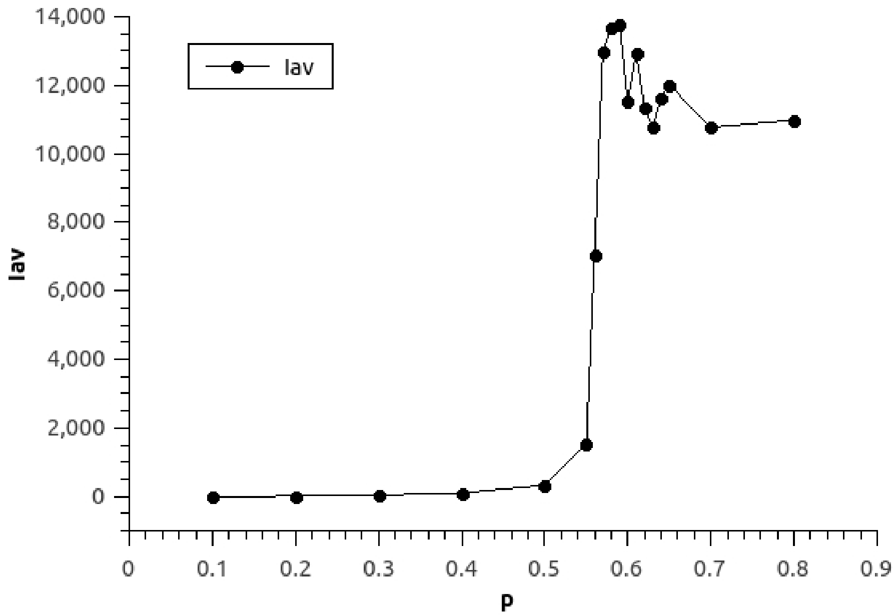
$$I'_{av}(p) = k'|p - p_c|^{-\gamma'}, p > p_c \tag{5}$$

$$P_\infty(p) = k^n|p - p_c|^{-\beta}, p < p_c \tag{6}$$

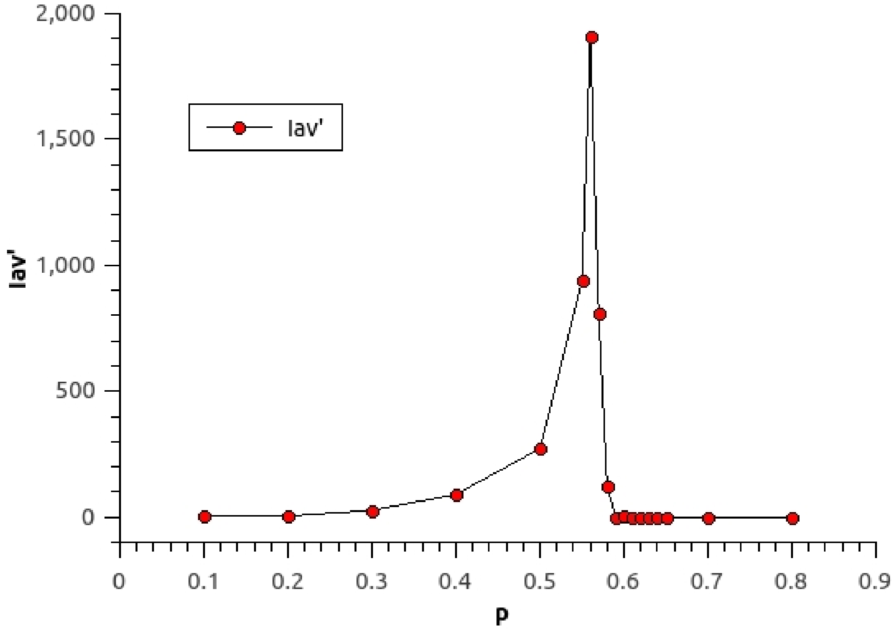
Αποτελέσματα

Παρακάτω δίνονται τα διαγράμματα των μέσων όρων των I_{av},I'_{av} και P_{max} για 100 επαναλήψεις και για ένα πλέγμα 200×200.

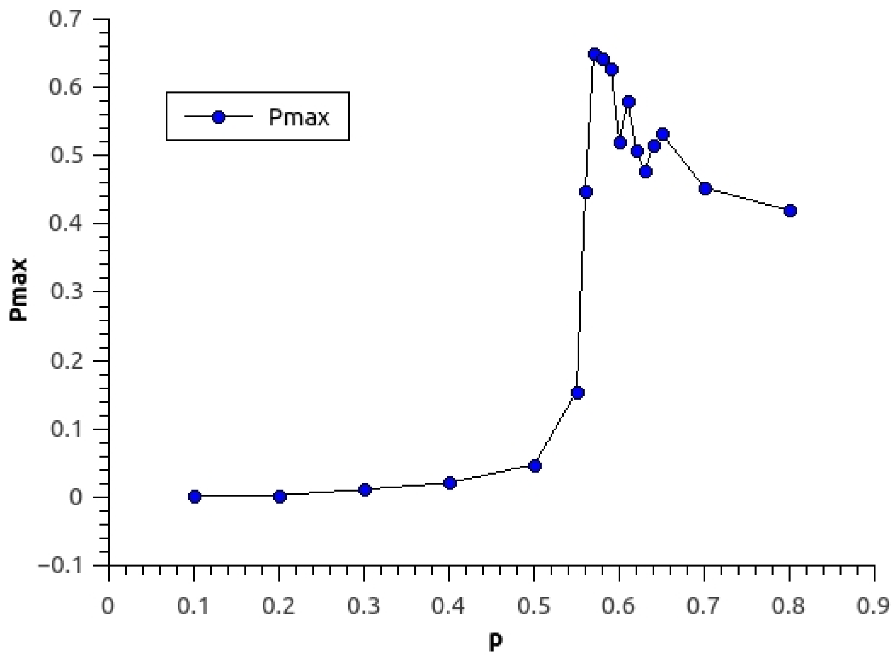
Σχήμα 1: $I_{av} - p$



Σχήμα 2: $I'_{av} - p$



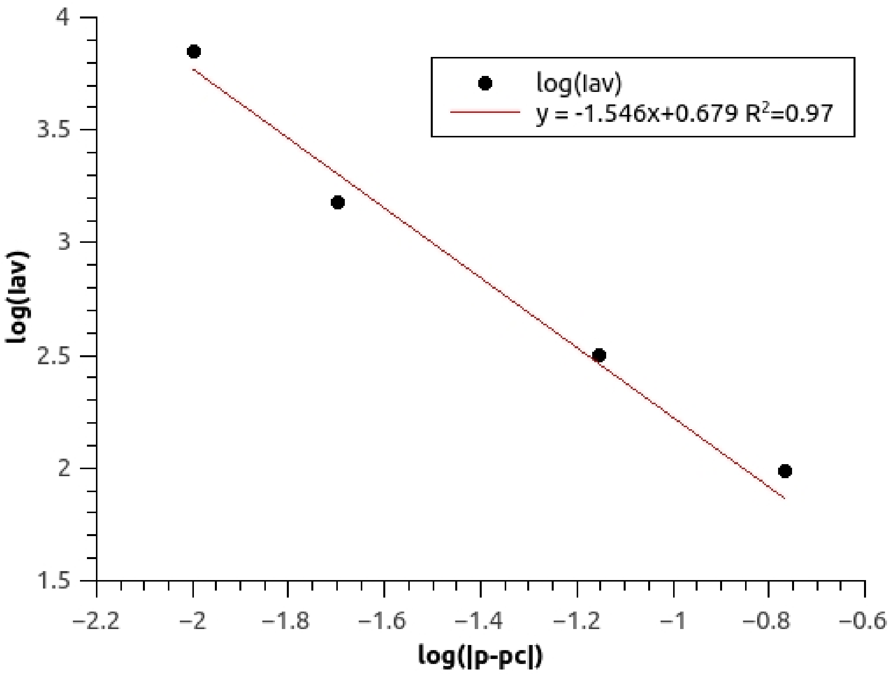
Σχήμα 3: $P_{max} - p$



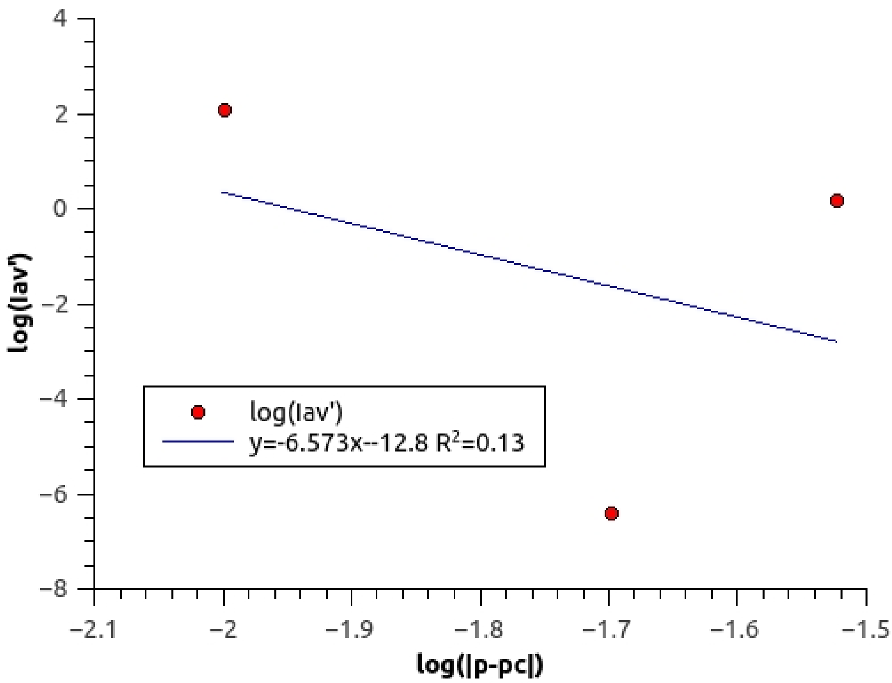
Από τα διαγράμματα βλέπουμε ότι η κρίσιμη πιθανότητα, η πιθανότητα δηλαδή που πρέπει να έχουν οι κλειστές θέσεις να εμφανιστούν έτσι ώστε να έχουμε διήθηση, είναι $\mathbf{p_c = 0.57}$.

Για αυτή τη κρίσιμη πιθανότητα κάνουμε τα διαγράμματα $I_{av} - |p - p_c|$, $I^{'}_{av} - |p - p_c|$, $P_{max} - |p - p_c|$ ώστε από την κλίση της ευθείας ελαχίστων τετραγώνων να βρούμε τους κρίσιμους εκθέτες $\beta, \gamma, \gamma^{'}$ σύμφωνα με τις σχέσεις (4), (5) και (6).

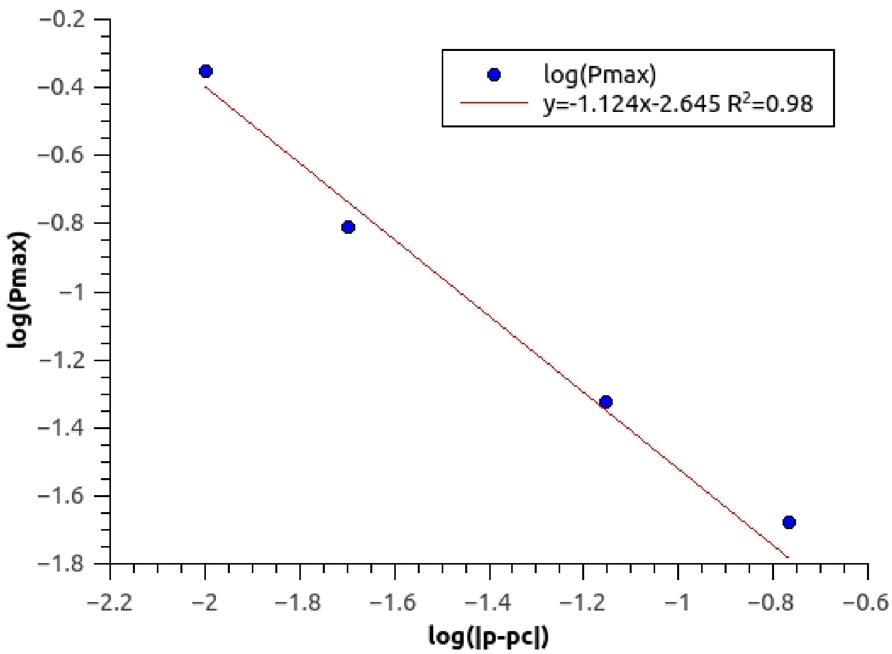
Σχήμα 4: $I_{av} - |p - p_c|$



Σχήμα 5: $I'_{av} - |p - p_c|$



Σχήμα 6: $P_{max} - |p - p_c|$



Άρα οι τιμές των κρίσιμων εκθετών είναι:

- $\gamma = 1.546$
- $\gamma^{'} = 6.573$
- $\beta = 1.124$

Από την τιμή των συντελεστών προσδιορισμού μπορούμε να πούμε ότι δεν εμπιστευόμαστε την τιμή του εκθέτη $\gamma^{'}$, ενώ έχουμε αρκετά ακριβή εκτίμηση για τα γ και β. Οι θεωρητικές τιμές των γ και β είναι: $\gamma = 43/18 = 2,38$ και $\beta = 5/36 = 0,138$.

Ο κώδικας που χρησιμοποιήσαμε για να υπολογίσουμε τα παραπάνω δίνεται στην επόμενη παράγραφο.

Ο κώδικας

```
#include <fstream>
#include <cstdlib>
#include <bits/stdc++.h>
#include <vector>
using namespace std;

int main(int argc, char const *argv[]) {
  int i,j,k,ii,l,m,temp,max;
  double temp2,count,sum1,sum2,sum3;
  double x;
  double p[] = {0.1,0.2,0.3,0.4,0.5,0.55,0.56,0.57,0.58,0.59, \
                0.6,0.61,0.62,0.63,0.64,0.65,0.7,0.8};
  int sizep = sizeof(p)/sizeof(p[0]);
  int n = 201;
  int N = 100;
  double Iav[sizep][N];
  double Iav2[sizep][N];
  double Pmax[sizep][N];
  vector<vector<int> > grid(n,vector<int> (n));
  int size = 100*n;
  int S[size];
  int L[size];

  for(i=0;i<size;i++){
    S[i] = 0;
    L[i] = 0;
  }

  srand(4372);
  ofstream f1("Iav.txt");
  ofstream f2("Iav2.txt");
  ofstream f3("Pmax.txt");

  for(m=0;m<N;m++){
```

```
    grid.assign(n, vector < int >(n, 0));
    for(l=0;l<sizep;l++){
      // clusters creation
      for(i=1;i<n;i++){
        for(j=1;j<n;j++){
          x = ((double) rand() / (RAND_MAX));
          if(x<p[l]){
            grid[i][j] = 1;
          }
        }
      }

      // CMLT
      k = 1;
      for(j=1;j<n;j++){
        for(i=1;i<n;i++){
          if(grid[i][j]==1){
            if(grid[i][j-1]==0){
              if(grid[i-1][j]==0){
                L[k] = k;
                grid[i][j] = L[k];
                S[L[grid[i][j]]]= S[L[grid[i][j]]] + 1;
                k = k + 1;
                continue;
              } else {
                grid[i][j] = grid[i-1][j];
                S[L[grid[i][j]]] = S[L[grid[i][j]]] + 1;
                continue;
              }
            } else {
              grid[i][j] = grid[i][j-1];
              S[L[grid[i][j]]] += 1;
              if(grid[i-1][j]==0 || grid[i-1][j]==grid[i][j-1]){
                continue;
              } else {
                temp = L[grid[i-1][j]];
                L[grid[i-1][j]] = L[grid[i][j-1]];
                for(ii=1;ii<size;ii++){
                  if(L[ii]==temp){
                    L[ii] = L[grid[i][j-1]];
                  }
                }
                S[L[grid[i][j-1]]] = S[L[grid[i][j-1]]] + S[grid[i-1][j
   ]];
                S[grid[i-1][j]] = 0;
                continue;
              }
            }
          }
```

```
        }
      }

      //maximum size
      max = S[1];

      for(i=2;i<size;i++){
        if(S[i]>max){
          max = S[i];
        }
      }

      // Iav
      count = 0;
      for(i=1;i<size;i++){
        temp2 = S[i]/(p[l]*n*n);
        count += temp2*S[i];
      }
      Iav[l][m] = count;

      //Iav'
      count = 0;
      for(i=1;i<size;i++){
        if(S[i]!=max){
          temp2 = S[i]/(p[l]*n*n);
          count += temp2*S[i];
        }
      }
      Iav2[l][m] = count;

      //Pmax
      Pmax[l][m] = max/(p[l]*n*n);

      for(i=0;i<size;i++){
        S[i] = 0;
        L[i] = 0;
      }
    }
  }

  for(i=0;i<sizep;i++){
    sum1 = 0;
    sum2 = 0;
    sum3 = 0;
    for(j=0;j<N;j++){
      sum1 += Iav[i][j];
      sum2 += Iav2[i][j];
      sum3 += Pmax[i][j];
    }
```

```
    f1 << p[i] << "␣" << sum1/N << endl;
    f2 << p[i] << "␣" << sum2/N << endl;
    f3 << p[i] << "␣" << sum3/N << endl;
  }

  return 0;
}
```