

Relazione Modellazione e Simulazioni Numeriche:
Percolazione

Melioli Riccardo e Bergonzani Ivan

Sessione Estiva 2015

Contents

1	Cenno Teorico	3
1.1	Introduzione alla Teoria della Percolazione	3
2	Esperimenti	4
2.1	Creazione del reticolo e Cluster Finding	4
2.1.1	Creazione del reticolo	4
2.1.2	Algoritmo Etichetta	4
2.1.3	Algoritmo Hoshel-Kopelman	7
2.2	Frequenza di percolazione e probabilità critica p_c	10
2.2.1	Descrizione e approccio intrapreso	10
2.2.2	Codice MATLAB	11
2.2.3	Risultati e osservazioni	12
2.3	R.A.C.S. (Reduced Average Cluster Size)	13
2.3.1	Descrizione e approccio intrapreso	13
2.3.2	Codice MATLAB	14
2.3.3	Risultati e osservazioni	15
2.4	P_1, P_2, P_3	18
2.4.1	Descrizione e approccio intrapreso	18
2.4.2	Codice MATLAB	19
2.4.3	Risultati e osservazioni	20
2.5	Confronto etichetta e HK	22
2.5.1	Descrizione e approccio intrapreso	22
2.5.2	Codice MATLAB	22
2.5.3	Risultati e osservazioni	25
3	Approfondimenti	28
3.1	Gestione dei bordi del reticolo nell'Algoritmo Etichetta	28
3.2	Ottimizzazione del algoritmo HK tramite calcolo parallelo . .	31
3.2.1	Descrizione e approccio intrapreso	31
3.2.2	Codice MATLAB	31
3.2.3	Risultati e osservazioni	35
3.3	Percolazione su reticolo 3D	35
3.3.1	Descrizione e approccio intrapreso	35

3.3.2	Codice MATLAB	36
3.3.3	Frequenza di percolazione su reticolo 3D	39
3.3.4	Codice MATLAB frequenza di percolazione 3D	40
3.3.5	Osservazioni e Differenze tra percolazione 2D e 3D . .	41

Chapter 1

Cenno Teorico

1.1 Introduzione alla Teoria della Percolazione

La Teoria della percolazione studia il problema della connettività tra elementi appartenenti ad un sistema, disposti nello spazio secondo una distribuzione definita a priori. Due elementi si possono definire connessi se sono adiacenti (relazione a corta distanza); Un insieme di elementi tra di loro connessi ricorsivamente forma un cluster.

Un obiettivo della teoria della percolazione è verificare la presenza all'interno di un sistema di uno o più cluster "Percolanti" ovvero di dimensioni comparabili a quelle del sistema stesso.

Una esempio classico del problema della percolazione è quello di una pietra porosa sul quale si versa dell'acqua verificando se quest'ultima, passando attraverso la pietra, la attraversi completamente (presenza di cluster percolante, ovvero di passaggio attraverso la pietra).

Idealmente vorremo studiare sistemi di dimensioni infinite ma non essendo possibile ci limitiamo a sistemi chiusi rappresentati da reticoli in d -dimensioni (Nel nostro caso $d=2$).

Chapter 2

Esperimenti

2.1 Creazione del reticolo e Cluster Finding

Algoritmi di base utilizzati per il conseguimento degli esperimenti.

2.1.1 Creazione del reticolo

Algoritmo utilizzato per la creazione del reticolo di lato L ("dim") e per la colorazione dei siti con probabilità "Prob_Col". Il reticolo creato(matrice LxL) avrà L^2 siti verrà ritornato come risultato della chiamata alla funzione.

```
% CreaCol_Ret.m
function [Reticolo_Col] = CreaCol_Ret (dim , Prob_Col)
    % Creazione e colorazione del reticolo con lato di dimensione "dim" con
    % probabilita di colorazione "Prob_Col"
    Reticolo_Col = rand(dim) < Prob_Col;
end
```

2.1.2 Algoritmo Etichetta

Algoritmo utilizzato per la ricerca dei cluster appartenenti a "Reticolo_Col" e per la numerazione dei siti appartenenti.

Questo algoritmo inizia da un sito occupato di "Reticolo_Col" e identifica l'intero cluster a cui questo sito appartiene, visitando, di volta in volta, i primi vicini(sinistro, superiore, destro e inferiore) dell'ultimo sito visitato; durante l'analisi vengono etichettati i siti già visitati così da non visitarli ulteriormente.

L'algoritmo utilizza principalmente due strutture dati: un coda ("Coda_Clus") in cui verranno inseriti i riferimenti ai siti appartenenti al cluster che si sta visitando e una matrice ("Reticolo_AE", delle stesse dimensioni di "Reticolo_Col") in cui ogni sito conterrà l'etichetta identificativa del cluster di

appartenenza. La matrice "Reticolo_AE" viene inoltre utilizzata per verificare se un sito fosse già stato visitato. Salviamo le dimensioni dei cluster appartenenti a "Reticolo_Col" all'interno di "Dim_Clus".

Per maggiore leggibilità del codice, la visita di ogni cluster è svolta da una funzione ("Vis_Clus_E_MEL") a sé stante. La funzione riceve come input una coda ("Coda_Clus") contenente il riferimento al sito del cluster da cui iniziare la visita. In output la funzione ritornerà "Coda_Clus" (dopo aver inserito i riferimenti agli altri siti del cluster) insieme al Reticolo_AE (passato anch'esso in input) modificato opportunamente.

```
% Alg_Etichetta_MEL.m
function [Reticolo_AE, Dim_Clus] = Alg_Etichetta_MEL (Reticolo_Col)
% Algoritmo Etichetta : Ricerca e numerazione dei cluster in un
% reticolo colorato, cio associa ad ogni sito il cluster di appartenenza.
Dim_Clus = 0;
Num_Clus=0;
Reticolo_AE=ones(size(Reticolo_Col))*-1;
% Metodo alternativo ottimizzante: consiste nel contornare temporaneamente
% i reticoli con siti non significativi(=0) per evitare i controlli dei siti
% adiacenti( fuori dal range) a quelli disposti sui bordi
Reticolo_Col = [zeros(1,size(Reticolo_Col,2)); Reticolo_Col;
zeros(1,size(Reticolo_Col,2))];
Reticolo_Col = [zeros(size(Reticolo_Col,1),1), Reticolo_Col, ...
zeros(size(Reticolo_Col,1),1)];
Reticolo_AE = [zeros(1,size(Reticolo_AE,2)); Reticolo_AE;
zeros(1,size(Reticolo_AE,2))];
Reticolo_AE = [zeros(size(Reticolo_AE,1),1), Reticolo_AE, ...
zeros(size(Reticolo_AE,1),1)];

% per ogni colonna del Reticolo_Col
for col = 2:size(Reticolo_Col)-1
%per ogni elemento di colonna del reticolo_col
for rig = 2:size(Reticolo_Col)-1
% se Reticolo_Col    occupato e non visitato aggiungiamo alla coda
% e visitiamo cluster di appartenenza
    if (Reticolo_AE(rig,col) == -1)
        if (Reticolo_Col(rig,col) == 1)
            Num_Clus=Num_Clus+1;
            Coda_Cluster = [rig col];
            Reticolo_AE(rig,col) = Num_Clus;
            [Reticolo_AE,Coda_Cluster]=Vis_Clus_E_MEL(Reticolo_Col, ...
            ... Reticolo_AE, Coda_Cluster, Num_Clus);
            % Salviamo la dimensione dell'ultimo cluster trovato
            Dim_Clus(Num_Clus+1) = size(Coda_Cluster,1);
            %finito di visitare il cluster si procede con gli altri siti
        else
            Reticolo_AE(rig,col) = 0;
        end
    end
end
end
end
```

```

% rimozione dei siti non significativi(=0)
Reticolo_Col = Reticolo_Col(2:size(Reticolo_Col,1)-1,2:size(Reticolo_Col,2)-1);
Reticolo_AE = Reticolo_AE(2:size(Reticolo_AE,1)-1,2:size(Reticolo_AE,2)-1);
Dim_Clus(1) = [];
end

```

```

Vis_Clus_E_MEL.m
function [Reticolo_AE,Coda_Cluster] = Vis_Clus_E_MEL(Reticolo_Col, Reticolo_AE, Coda_Clus)
% indice massimo per le operazioni sui reticoli
L = size(Reticolo_Col,1);

i=0;
% ciclo in cui scorriamo l'intera coda cosi da visitare i nodi a cui fanno
% riferimento gli indici contenuti in essa
while i < size(Coda_Cluster,1)
    i = i+1;
    % controllo dei siti adiacenti in senso orario (sopra,destra...)
    % indici elemento sopra il sito centrale
    rig = Coda_Cluster(i,1)-1;
    col = Coda_Cluster(i,2);
    if (Reticolo_AE(rig,col) == -1)
        if (Reticolo_Col(rig,col) == 1)
            % accodiamo a Coda_Cluster gli indici del sito visitato
            Coda_Cluster = [Coda_Cluster; rig, col];
            % Identifichiamo il sito con il numero del cluster di
            % appartenenza
            Reticolo_AE(rig,col) = Num_Clus;
        else
            Reticolo_AE(rig,col) = 0;
        end
    end

    % indici elemento destro del sito centrale
    rig = Coda_Cluster(i,1);
    col = Coda_Cluster(i,2)+1;
    if (Reticolo_AE(rig,col) == -1)
        if (Reticolo_Col(rig,col) == 1)
            % accodiamo a Coda_Cluster gli indici del sito visitato
            Coda_Cluster = [Coda_Cluster; rig, col];
            % Identifichiamo il sito con il numero del cluster di
            % appartenenza
            Reticolo_AE(rig,col) = Num_Clus;
        else
            Reticolo_AE(rig,col) = 0;
        end
    end

    % indici elemento inferiore al sito centrale
    rig = Coda_Cluster(i,1)+1;
    col = Coda_Cluster(i,2);

```

```

if (Reticolo_AE(rig,col) == -1)
    if (Reticolo_Col(rig,col) == 1)
        % accodiamo a Coda_Cluster gli indici del sito visitato
        Coda_Cluster = [Coda_Cluster; rig, col];
        % Identifichiamo il sito con il numero del cluster di
        % appartenenza
        Reticolo_AE(rig,col) = Num_Clus;
    else
        Reticolo_AE(rig,col) = 0;
    end
end

end

% indici elemento sinistro al sito centrale
rig = Coda_Cluster(i,1);
col = Coda_Cluster(i,2)-1;
if (Reticolo_AE(rig,col) == -1)
    if (Reticolo_Col(rig,col) == 1)
        % accodiamo a Coda_Cluster gli indici del sito visitato
        Coda_Cluster = [Coda_Cluster; rig, col];
        % Identifichiamo il sito con il numero del cluster di
        % appartenenza
        Reticolo_AE(rig,col) = Num_Clus;
    else
        Reticolo_AE(rig,col) = 0;
    end
end

end

end
end

```

2.1.3 Algoritmo Hoshel-Kopelman

Un altro algoritmo per la ricerca di cluster(cluster finding) è l'algoritmo di Hoshen-Kopelman, abbreviato "HK".

La visita del reticolo ("Reticolo_Col", preso in input) viene effettuata sito per sito, procedendo per colonne, iniziando dal sito posto in alto a sinistra fino ad arrivare a quello in basso a destra.

Le dimensioni e le relazioni dei cluster vengono memorizzate in un array ("DimRef.Label"), cioè ad ogni sito visitato viene assegnata come etichetta la minore tra quella del sito vicino superiore e quella del sito vicino sinistro(non considerando valori che non facciano riferimento a etichette, esempio: valore indicante un sito vuoto); nel caso in cui i siti adiacenti sopraccitati siano entrambi vuoti(non colorati), il sito di riferimento apparterrà a un nuovo cluster a cui verrà assegnato una nuova etichetta.

Le dimensioni dei cluster vengono memorizzate in un array ("DimRef.Label"). Quando il cluster preso in considerazione è un sottoinsieme di un cluster adiacente, viene accorpato, sommando la sua dimensione a quella del cluster adiacente e memorizzandola in quest'ultimo; infine nel cluster accorpato, invece che la dimensione, verrà memorizzato il riferimento al cluster accor-

pante.

L'algoritmo restituisce in output un reticolo in cui ogni sito colorato etichettato in base al cluster di appartenenza e un array contenente le dimensioni dei cluster trovati.

```
%Alg_HK.m
function [Reticolo_AHK, Dim_Clus] = Alg_HK (Reticolo_Col)
% Algoritmo HK : Ricerca e numerazione dei cluster in un
% reticolo colorato, cio associa ad ogni sito il cluster di appartenenza.
[r c] = size(Reticolo_Col);
Dim_Clus = [];
Num_Clus= 0;
Reticolo_AHK= zeros(r,c);

% lista delle relazioni tra cluster adiacenti
DimRef_Label( 1:ceil((r*c)/2) ) = 0;

% per ogni colonna del Reticolo_Col
for col = 1:c
    %per ogni elemento di colonna del reticolo_col
    for rig = 1:r
        % se Reticolo_Col    occupato/colorato
        if (Reticolo_Col(rig,col) == 1)
            % verifichiamo in che situazione ci troviamo tenendo conto
            % anche della situazione agli estremi
            Situaz = 0;

            % se non siamo sulla prima colonna controlliamo solo il vicino
            % sinistro
            if col > 1
                % Etichette degli elemento adiacenti sinistro
                Label_Sx = Reticolo_AHK(rig,col-1);
                Situaz = sign(Label_Sx);
            end
            % se non siamo sulla prima riga controlliamo solo il vicino
            % superiore
            if rig > 1
                % Etichette degli elemento adiacenti superiore
                Label_Up = Reticolo_AHK(rig-1,col);
                Situaz = Situaz + 2*sign(Label_Up);
            end

            switch Situaz
                % Situaz=0 -> vicini superiore e sinistro non colorati
                case 0
                    % nuovo cluster. Aumentiamo il numero di cluster
                    % trovati nel reticolo
                    Num_Clus = Num_Clus+1;
                    Reticolo_AHK(rig,col) = Num_Clus;
                    DimRef_Label(Num_Clus) = DimRef_Label(Num_Clus) + 1;

                % Situaz=1 -> vicini sinistro colorato
```

```

case 1
    ref_ok = Label.Sx;
    while DimRef_Label(ref_ok) < 0
        ref_ok = -DimRef_Label(ref_ok);
    end
    % incrementiamo il contatore del cluster a cui fa
    % riferimento il vicino
    DimRef_Label(ref_ok) = DimRef_Label(ref_ok) + 1;
    % etichettiamo direttamente col riferimento
    Reticolo_AHK(rig,col) = ref_ok;

% Situaz=2 -> vicini superiore colorato
case 2
    ref_ok = Label.Up;
    while DimRef_Label(ref_ok) < 0
        ref_ok = -DimRef_Label(ref_ok);
    end
    % incrementiamo il contatore del cluster a cui fa
    % riferimento il vicino
    DimRef_Label(ref_ok) = DimRef_Label(ref_ok) + 1;
    % etichettiamo direttamente col riferimento
    Reticolo_AHK(rig,col) = ref_ok;

% Situaz=3 -> vicini superiore e sinistro colorati
case 3
    % cerchiamo ricorsivamente l'etichetta dei cluster
    % a cui fanno riferimento i vicini
    ref_up = Label.Up;
    ref_sx = Label.Sx;
    while DimRef_Label(ref_up) < 0
        ref_up = -DimRef_Label(ref_up);
    end
    while DimRef_Label(ref_sx) < 0
        ref_sx = -DimRef_Label(ref_sx);
    end

    % valutazioni sulle etichette trovate nei while
    if ref_sx < ref_up % ref SX < ref UP
        % coloriamo il sito con l'etichetta minore (sx)
        Reticolo_AHK(rig,col) = ref_sx;
        % incrementiamo il contatore del cluster della
        % etichetta minore con il valore del contatore
        % del cluster dell'etichetta maggiore
        DimRef_Label(ref_sx) = DimRef_Label(ref_sx) + ...
        ... DimRef_Label(ref_up) + 1;
        % il contatore dei cluster diventer quello del
        % cluster con etichetta minore tra i due
        DimRef_Label(ref_up) = -ref_sx;

    elseif ref_sx > ref_up % ref SX > ref UP
        Reticolo_AHK(rig,col) = ref_up;
        DimRef_Label(ref_up) = DimRef_Label(ref_up) + ...
        ... DimRef_Label(ref_sx) + 1;
        DimRef_Label(ref_sx) = -ref_up;
    end
end

```

```

elseif ref_sx == ref_up % SX = UP
    % coloriamo il sito con l'etichetta del cluster
    % di riferimento
    Reticolo_AHK(rig,col) = ref_up;
    % incrementiamo il contatore del cluster
    % di riferimento
    DimRef_Label(ref_up) = DimRef_Label(ref_up) + 1;
end
end
end
end
end
% Manteniamo solo le dimensione del vettore DimRef_Label (eliminando i riferimenti)
Dim_Clus = DimRef_Label;
Dim_Clus(DimRef_Label < 1) = [];
end

```

2.2 Frequenza di percolazione e probabilità critica

p_c

2.2.1 Descrizione e approccio intrapreso

L'obiettivo dell'esperimento è determinare la soglia di percolazione $p=p_c$ (con p corrispondente alla probabilità di colorazione dei siti del reticolo) oltre il quale si ha percolazione.

Questo indice viene stimato attraverso una serie di tentativi, su reticoli di diverse dimensioni, a cui associeremo per ogni probabilità p una frequenza ν_{perc} .

$$\nu_{perc} = \frac{N_{perc}}{N_{tentativi}}$$

All'aumentare del numero di tentativi la frequenza ν_{perc} convergerà alla probabilità P_{perc} di avere percolazione.

$$\lim_{N_{tentativi} \rightarrow \infty} \nu_{perc} = P_{perc}$$

La percolazione critica p_c viene stimata osservando la variazione dell'andamento del grafico delle frequenze ν_{perc} per le diverse dimensioni del reticolo.

Per ogni valore calcolato di ν_{perc} viene considerato anche l'errore commesso a causa dal limitato numero di prove effettuate.

$$\Delta \nu_{perc} = \frac{\sigma}{\sqrt{N_{tentativi}}}$$

dove σ è la deviazione standard:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N_{tentativi}} (xi - \bar{x})^2}{N_{tentativi}}}$$

in cui \bar{x} è la media aritmetica.

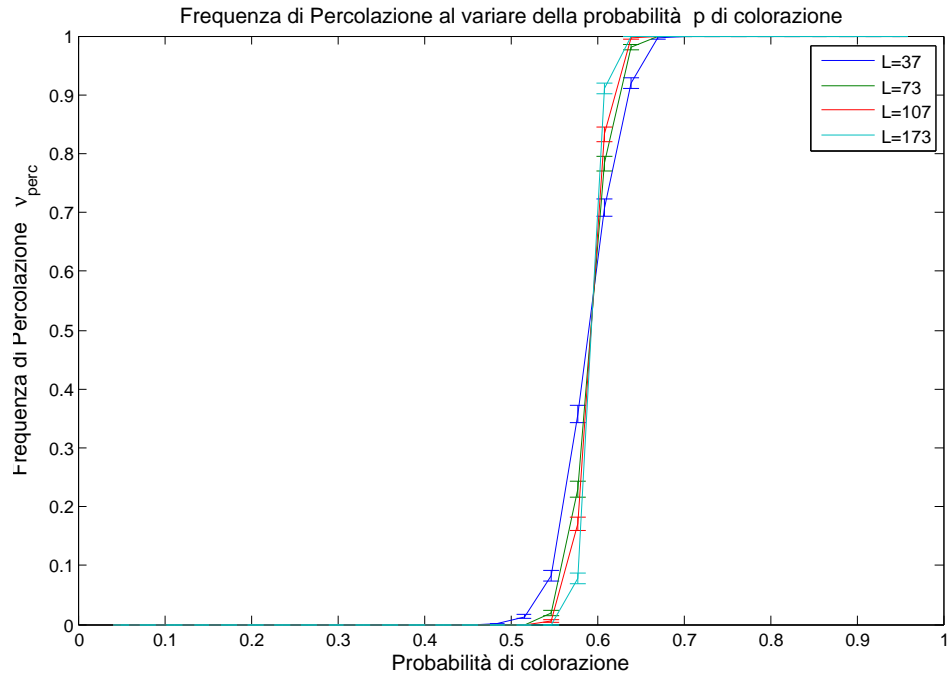
Idealmente un numero infinito di prove ci permetterebbe di avere una stima non affetta da errore.

2.2.2 Codice MATLAB

```
% Test sulla Frequenza di Percolazione (Prove in base alla probabilit
% eseguite in parallelo)
clc
clear
% numero probabilit
np=30;
% probabilita nel range [0.05 0.95]
np_range=linspace(0.05,0.95,np);
% numero di tentativi per ogni probabilit
tent = 1000;
% dimensione reticolo
for dim=[37, 73, 107, 173]
    Freq_Perc = zeros(1,np);
    % numero di tentativi parallelizzati
    parfor i=1:np
        % probabilit di colorazione
        p = np_range(i);
        perc = zeros(1,tent);
        % i casi limite con 0 e 1 sappiamo che descivono rispettivamente
        % l'assenza di percolazione e la presenza certa
        for t = 1 : tent
            Reticolo=CreaColRet(dim,p);
            [Reticolo_AE, Dim_Clus] = Alg_Etichetta_MEL(Reticolo);
            if Ricer_Percol(Reticolo_AE) ~= 0
                Freq_Perc(i) = Freq_Perc(i) + 1;
                perc(t) = 1;
            end
        end
        % CALCOLO FREQUENZA DI PERCOLAZIONE (vPerc)
        Freq_Perc(i) = Freq_Perc(i)/tent;
        % CALCOLO DELL'ERRORE (associato ad ogni probabilit p)
        % mediante funzione matlab 'std' che ritorna la deviazione standard
        Err_Freq_Perc(i) = std(perc)/sqrt(tent);
    end
    % GRAFICO
    hold all
    errorbar(np_range,Freq_Perc,Err_Freq_Perc);
end
hold off
```

2.2.3 Risultati e osservazioni

I seguenti grafici rappresentano i risultati ottenuti dai test sulla frequenza di percolazione.



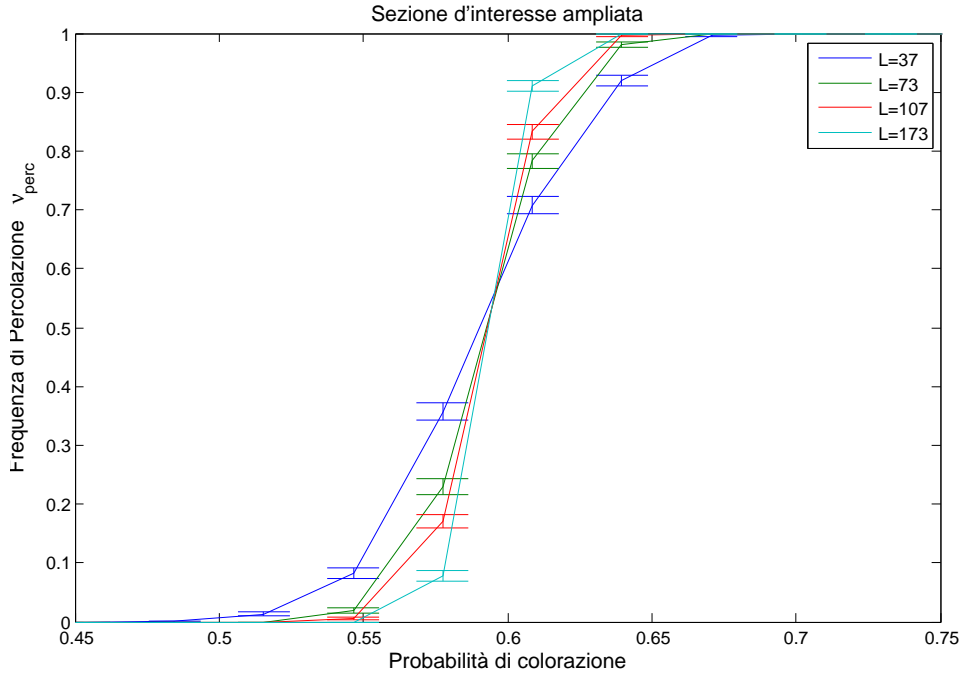


Figure 2.1: Test sulla frequenza di percolazione, 1000 tentativi

Osservando i grafici di figura 2.1 possiamo desumere che, all'aumentare della dimensione del reticolo, la funzione risulti sempre più rapida nel passaggio dalla probabilità di colorazione con assenza certa di percolazione alla probabilità dove vi è presenza certa. Possiamo perciò concludere che per un reticolo di dimensione infinita il passaggio da assenza a presenza certa di percolazione risulti senza valori di probabilità intermedi (netto); questo passaggio si assesta in un intorno del valore di probabilità $p=0.6$ che risulta essere il valore della soglia di percolazione cercato.

2.3 R.A.C.S. (Reduced Average Cluster Size)

2.3.1 Descrizione e approccio intrapreso

Introduciamo ora il R.A.C.S. (Reduced Average Cluster Size), quantità che rappresenta, attraverso una media pesata, la dimensione media dei cluster di un reticolo senza però considerare quello di dimensione maggiore. Il R.A.C.S. è definito come:

$$R.A.C.S. = \frac{\sum_{s \neq s_{max}} s(s \cdot n_s)}{\sum_s s \cdot n_s}$$

con

- s = dimensione del cluster
- n_s = numero di cluster di dimensione s
- S_{max} = dimensione del cluster massimo

Consideriamo per ogni valore R.A.C.S. calcolato anche il relativo errore:

$$\Delta_{R.A.C.S.} = \frac{\sigma}{\sqrt{N_{tentativi}}}$$

dove σ è la deviazione standard:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N_{tentativi}} (x_i - \bar{x})^2}{N_{tentativi}}}$$

in cui \bar{x} è la media aritmetica.

L'obiettivo dell'esperimento è analizzare l'andamento del R.A.C.S. in base alla probabilità p di colorazione dei siti di reticoli di diverse dimensioni.

2.3.2 Codice MATLAB

```
% RACS TEST CON PROVE IN PARALLELO
% numero probabilita
np=30;
% probabilita nel range [0.01 1]
np_range=linspace(0.01,1,np);
% numero di tentativi per ogni probabilita
tent = 1000;

% dimensione reticolo
for dim=[ 37, 73, 107, 173]
    % numero di tentativi parallelizzati
    parfor i=1:np
        p = np_range(i);
        RACS = zeros(1,tent);
        % i casi limite con 0 e 1 sappiamo che descrivono rispettivamente
        % l'assenza di percolazione e la presenza
        for t = 1 : tent
            Reticolo=CreaColRet(dim,p);
            [Reticolo_AHK, Dim_Clus] = Alg.HK(Reticolo);
            % tmp Dim_Clus senza il valore massimo
            tmp = Dim_Clus;
            [valmax,index] = max(Dim_Clus);
            tmp(index) = [];
            % Racs sulla singola prova
            RACS(t) = sum(tmp.^2)/sum(Dim_Clus);
        end
    end
    % MEDIA del RACS e relativo ERRORE per ogni probabilita
```

```

        Avg_RACS(i) = mean(RACS);
        Err_RACS(i) = std(RACS)/sqrt(tent);
    end
    % GRAFICO
    hold all
    errorbar(np_range,Avg_RACS,Err_RACS,'-');
end

```

```

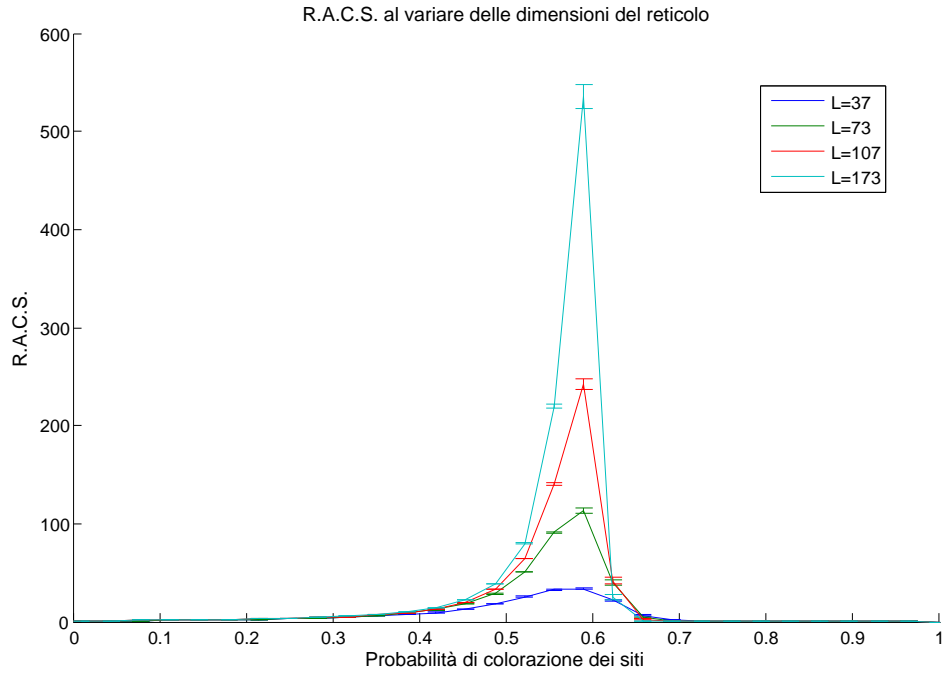
TestImmaginiPerc.m
% numero probabilit per ogni dimensione
np=8;
% probabilita nel range [0.2 0.9]
np_range=linspace(0.2,0.9,np);

for zi=1:np
    % probabilit percolazione
    p = np_range(zi);
    Reticolo=CreaCol.Ret(30,p);
    [Reticolo_AE, Dim_Clus] = Alg.Etichetta_MEL(Reticolo);
    [Clus_perc] = Ricer.Percol(Reticolo_AE);
    Reticolo = double(Reticolo);
    for i= 1:size(Reticolo,1)
        for j= 1:size(Reticolo,1)
            if (Reticolo_AE(i,j) == Clus_perc)
                Reticolo(i,j) = 0.5;
            end
        end
    end
    Reticolo(30,30) = 1;
    subplot(3,3,zi)
    surf(Reticolo);
    view(2)
end

```

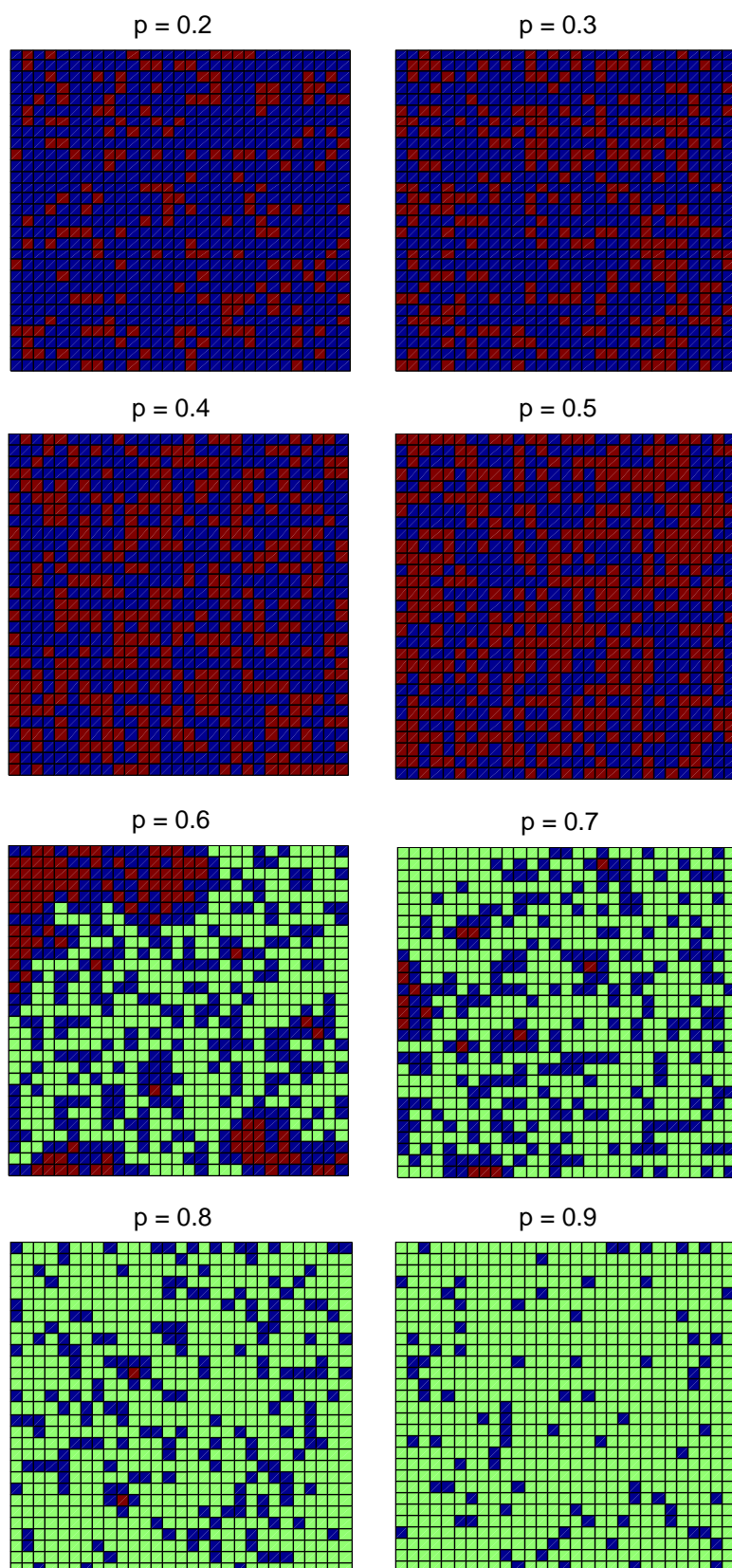
2.3.3 Risultati e osservazioni

Il seguente grafico rappresentano i risultati ottenuti dai test sull'indice R.A.C.S.

Figure 2.2: R.A.C.S., $L = 37, 73, 107, 173, 1000$ prove

Osservando il grafico di figura 2.2 possiamo desumere che oltre la soglia di percolazione, maggiormente all'aumentare della dimensione del reticolo, si ha un drastico calo della dimensione media dei cluster (senza considerare il cluster di dimensione maggiore).

Ciò accade perchè oltre a questa soglia, all'aumentare della probabilità di colorazione p , il cluster di dimensione maggiore (probabilmente percolante) ingloberà (da non intendere come evoluzione del reticolo) velocemente i cluster limitrofi comportando, dato che è considerato solo al denominatore, ad un calo della dimensione media dei cluster restanti. Il fenomeno finora descritto rappresentato dalla seguente successione di figure illustranti un reticolo 30×30 colorato al crescere della probabilità p .

Figure 2.3: Colorazione cluster al variare di p , $L = 30$

Dalle precedenti figure possiamo osservare che con un valore p di circa 0.3 sono presenti solo cluster di piccola dimensione; con probabilità p di circa 0.5 la dimensione media dei cluster è aumentata, aumentando di conseguenza il R.A.C.S.

Oltre alla soglia di percolazione, quindi con valori di p maggiori a 0.6 (circa), possiamo notare che il cluster "percolante" abbia dimensione molto maggiore rispetto ai cluster adiacenti, causando un drastico calo del R.A.C.S.

2.4 P_1, P_2, P_3

2.4.1 Descrizione e approccio intrapreso

Introduciamo tre quantità relative ad un reticolo colorato con probabilità p :

P_1 quantità indicante la probabilità con cui un sito qualsiasi possa appartenere al cluster di dimensione massima, ed è definita da:

$$P_1 = \frac{S_{max}}{L^2}$$

P_2 quantità indicante una **stima** della probabilità con cui un sito colorato possa appartenere al cluster di dimensione massima. P_2 è definita dalla seguente formula:

$$P_2 = \frac{S_{max}}{p \cdot L^2}$$

P_3 quantità indicante l'**effettiva** probabilità con cui un sito colorato possa appartenere al cluster di dimensione massima S_{max} . P_3 è definita dalla seguente formula:

$$P_3 = \frac{S_{max}}{\sum_s s \cdot n_s}$$

con:

- S_{max} : Dimensione del massimo cluster del reticolo.
- L^2 : Dimensione del lato del reticolo al quadrato
- $p \cdot L^2$: Stima del numero di siti colorati

- s : taglia del cluster.
- n_s : indica il numero di cluster di dimensione s .

L'esperimento analizzerà i valori medi (in base al numero di tentativi) delle quantità sopra presentate al variare della probabilità e su reticoli di differenti dimensioni.

Consideriamo per ogni valore medio $P_{1,2,3}$ calcolato anche il relativo errore:

$$\Delta P_{1,2,3} = \frac{\sigma}{\sqrt{N_{tentativi}}}$$

dove σ è la deviazione standard:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N_{tentativi}} (x_i - \bar{x})^2}{N_{tentativi}}}$$

in cui \bar{x} è la media aritmetica.

2.4.2 Codice MATLAB

```
% P1, P2, P3 TEST CON PROVE IN PARALLELO

% numero probabilita
np=30;
% probabilita nel range [0.01 1]
np_range=linspace(0.01,1,np);
% numero di tentativi per ogni probabilita
tent = 1000;

% dimensione reticolo
afor dim=[ 37, 73, 107, 173]
    % numero di tentativi parallelizzati
    parfor i=1:np
        p = np_range(i);
        p1 = zeros(1,tent);
        p2 = zeros(1,tent);
        p3 = zeros(1,tent);
        % i casi limite con 0 e 1 sappiamo che descrivono rispettivamente
        % l'assenza di percolazione e la presenza
        for t = 1 : tent
            Reticolo=CreaColRet(dim,p);
            [Reticolo_AHK, Dim_Clus] = Alg_HK(Reticolo);
            % P1, P2, P3 sulla singola prova
            p1(t) = max(Dim_Clus)/(dim^2);
            p2(t) = p1(t)/p;
            p3(t) = max(Dim_Clus)/sum(Dim_Clus);
```

```

end
% MEDIA di P1, P2, P3 e relativi ERRORI per ogni probabilit
Avg_P1(i) = mean(p1);
Err_P1(i) = std(p1)/sqrt(tent);
Avg_P2(i) = mean(p2);
Err_P2(i) = std(p2)/sqrt(tent);
Avg_P3(i) = mean(p3);
Err_P3(i) = std(p3)/sqrt(tent);
end
% GRAFICI
figure(1);
hold all
errorbar(np_range,Avg_P1,Err_P1,'-');
figure(2)
hold all
errorbar(np_range,Avg_P2,Err_P2,'-');
figure(3)
hold all
errorbar(np_range,Avg_P3,Err_P3,'-');
end

```

2.4.3 Risultati e osservazioni

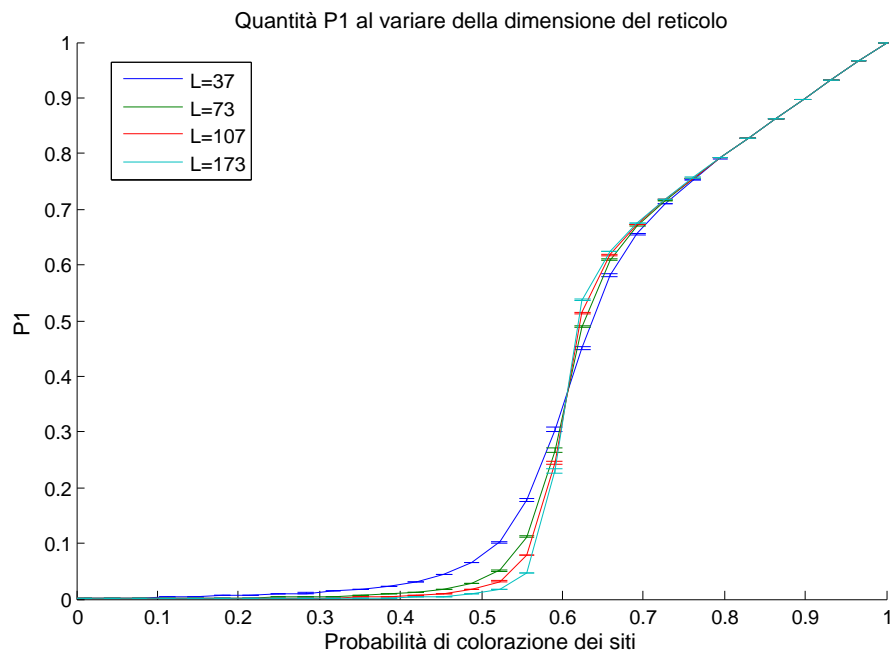
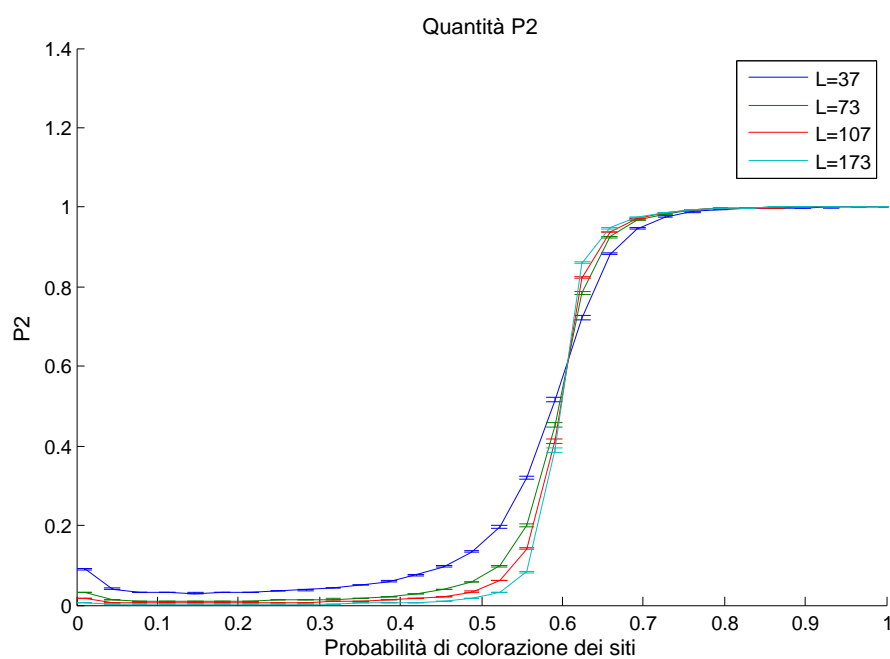
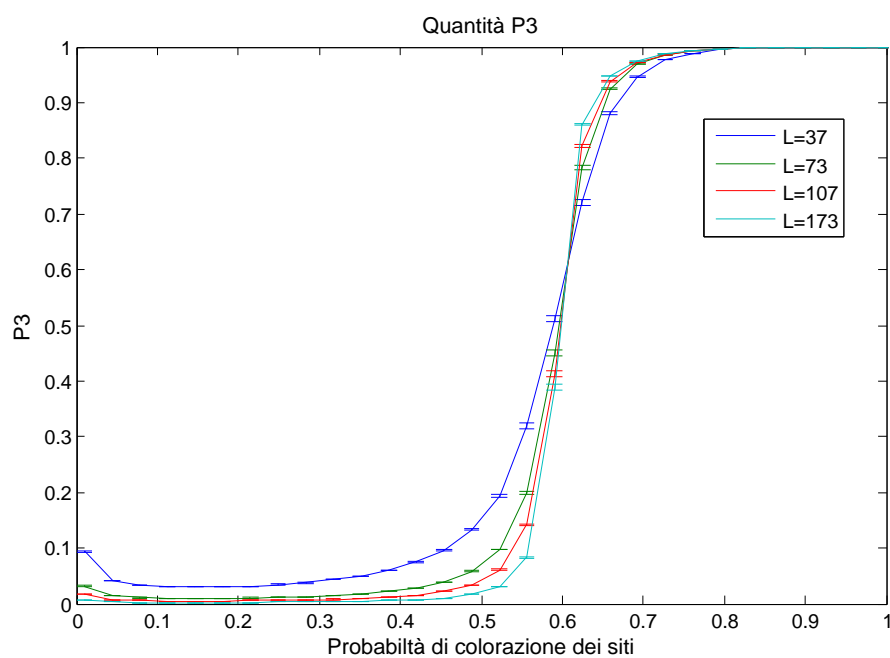


Figure 2.4: P_1 valutato con $L = 37, 73, 107, 173$, su 1000 tentativi

Figure 2.5: P_2 valutato con $L = 37, 73, 107, 173$, su 1000 tentativiFigure 2.6: P_3 valutato con $L = 37, 73, 107, 173$, su 1000 tentativi

Osservando i grafici delle figure 2.2, 2.3, 2.4 si può notare la rapida crescita di P_1 , P_2 , P_3 nell'intorno della soglia di percolazione perchè la dimensione del cluster massimo ha un picco della crescita in corrispondenza di questa soglia. All'aumentare della dimensione del reticolo l'ampiezza di quest'intorno diminuisce ed in particolare considerando un reticolo infinito la crescita avverrebbe in corrispondenza della soglia di percolazione; si deduce che il cluster S_{max} ingloberà più velocemente i cluster restanti all'aumentare della dimensione del reticolo ($L \rightarrow \infty$).

Inoltre nel grafico di figura 2.2 si osserva che intorno alla soglia di percolazione la quantità P_1 non raggiunge il valore 1, al contrario delle altre quantità, dato che vengono considerati tutti i siti del reticolo (anche quelli non colorati).

Dall'andamento del grafico di P_1 , nell'intervallo tra la soglia di percolazione e 1, si deduce come oltre tale soglia il cluster S_{max} ingloberà velocemente tutti gli altri cluster; la probabilità che un sito qualunque appartenga a S_{max} si avvicinerà quindi alla probabilità di questo sito di essere colorato che sappiamo essere strettamente connessa alla probabilità di colorazione dei siti.

2.5 Confronto etichetta e HK

2.5.1 Descrizione e approccio intrapreso

Questo Test confronta i tempi di esecuzione degli algoritmi di cluster labelling introdotti e utilizzati precedentemente. I tempi vengono ricavati inizialmente per diverse taglie del reticolo considerando per ogni linea probabilità di colorazione fissa e successivamente per diverse probabilità di colorazione considerando per ogni linea dimensione del reticolo fissa.

2.5.2 Codice MATLAB

```
% test di confronto tra i tempi di calcolo degli algoritmi Etichetta e HK

% numero probabilit
np=11;
% probabilita nel range [0 1]
np_range=linspace(0,1,np);
% numero di tentativi per ogni probabilit -dimensione
tent=17;
% dimensioni dei reticoli
dimensioni=[37, 73, 107, 137, 173, 199];

% Matrice dei tempi di esecuzione
% le righe indicano le dimensioni utilizzate
% le colonne indicano le probabilit utilizzate
```

```

% la profondita per i diversi tentativi
Tempi_AE = zeros(size(dimensioni,2), np, tent);
Tempi_HK = zeros(size(dimensioni,2), np, tent);
% Matrici dei tempi medi
Avg_Tempi_AE = zeros(size(dimensioni,2), np);
Avg_Tempi_HK = zeros(size(dimensioni,2), np);

% Ciclo sulle dimensioni del reticolo
for n=1:size(dimensioni,2)
    dim = dimensioni(1,n);
    % Ciclo sulle probabilit di colorazione (eseguito in parallelo)
    for i=1:np
        % probabilit di colorazione
        p = np_range(i);
        for t = 1 : tent
            reticolo = CreaCol_Ret(dim,p);
            tic; [ReticoloAE,NumClusAE] = Alg.Etichetta.MEL(reticolo); Tempi_AE(n,i,t) = toc;
            tic; [ReticoloHK,NumClusHK] = Alg.HK(reticolo); Tempi_HK(n,i,t) = toc;
        end
    end
end

% calcolo della media dei tempi (per ogni coppia probabilita-dimensione)
for d=1:size(dimensioni,2)
    for p=1:np
        Avg_Tempi_AE(d,p) = mean(Tempi_AE(d,p,:));
        Avg_Tempi_HK(d,p) = mean(Tempi_HK(d,p,:));
    end
end

f1 = figure;
f2 = figure;
f3 = figure;
f4 = figure;
% Grafici dei tempi di esecuzione a probabilit fissa (al variare della
% dimensione del reticolo)
% ALGORITMO ETICHETTA
figure(f1);
hold all
for p=1:np
    plot(dimensioni,Avg_Tempi_AE(:,p));
end
hold off
% ALGORITMO H-K
figure(f2);
hold all
for p=1:np
    plot(dimensioni,Avg_Tempi_HK(:,p));
end
hold off

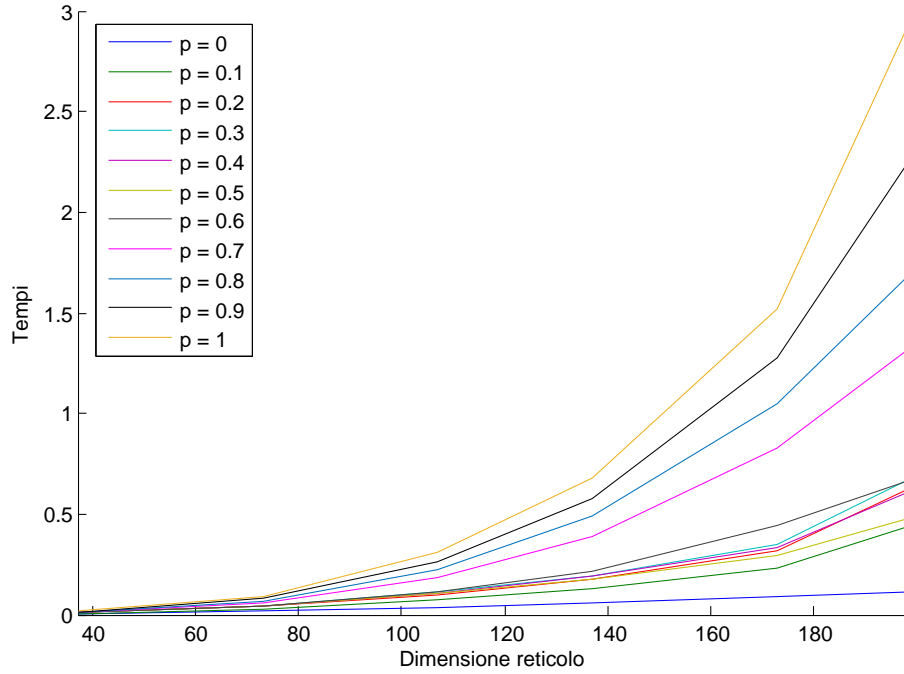
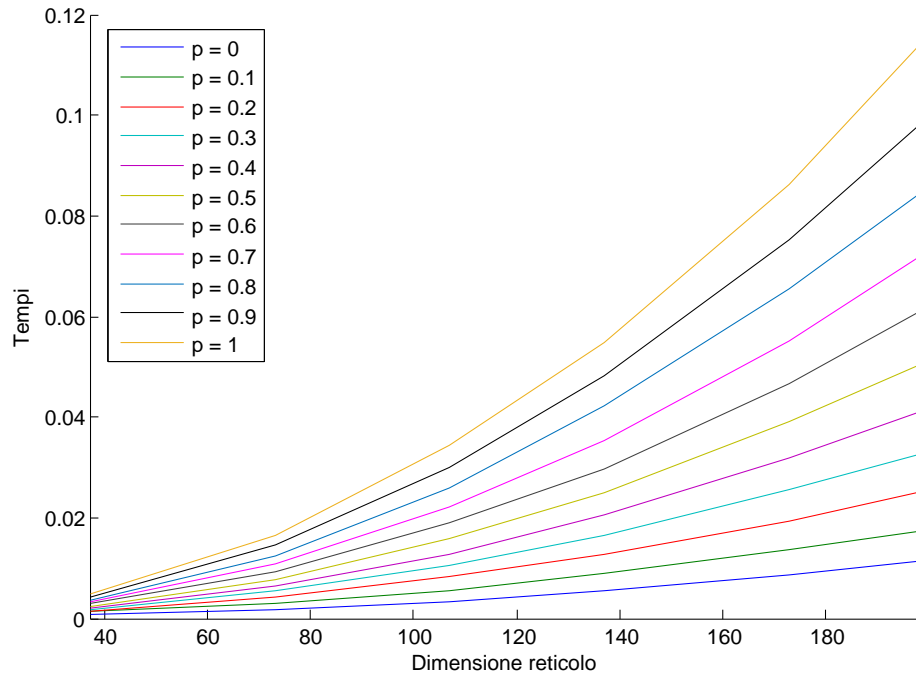
% Grafici dei tempi di esecuzione a dimensione fissa (al variare della
% probabilit di colorazione)
% ALGORITMO ETICHETTA
figure(f3);

```



```
hold all
for d=1:size(dimensioni,2)
    plot(np_range,Avg_Tempi_AE(d,:));
end
hold off
% ALGORITMO H-K
figure(f4);
hold all
for d=1:size(dimensioni,2)
    plot(np_range,Avg_Tempi_HK(d,:));
end
hold off
```

2.5.3 Risultati e osservazioni

Figure 2.7: Tempo di esecuzione algoritmo "Etichetta" al variare di L Figure 2.8: Tempo di esecuzione algoritmo "HK" al variare di L

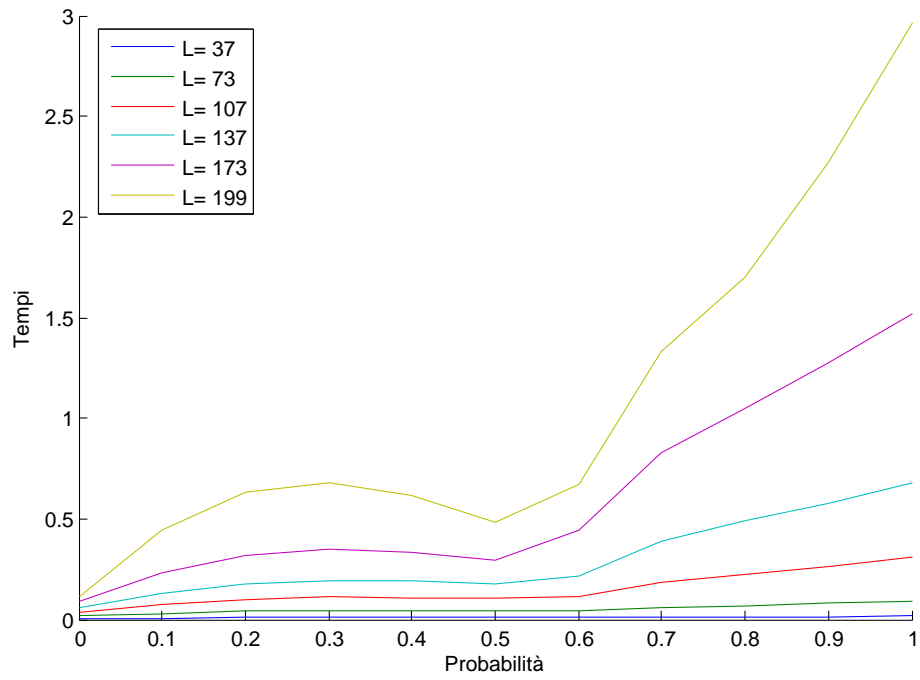


Figure 2.9: Tempo di esecuzione algoritmo "Etichette" al variare della probabilità

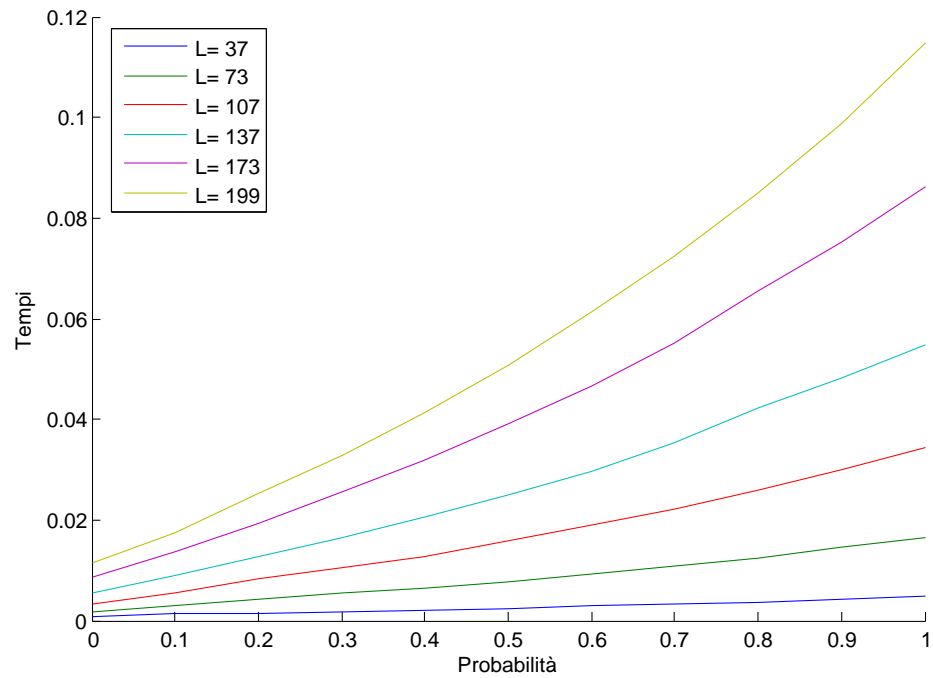


Figure 2.10: Tempo di esecuzione algoritmo "HK" al variare della probabilità

Da i risultati ottenuti si può denotare che l'algoritmo "HK" ha tempi di esecuzione molto minori. Quest'ultimo algoritmo, avendo un ordine preciso di visita dei siti, non si deve preoccupare di verificare se ognuno di essi sia già stato visitato e inoltre per ogni sito colorato controlla solo il vicino superiore e sinistro (non tutti e quattro): ne consegue un minore numero di accessi alla memoria.

Questo è permesso dalla gestione alternativa delle etichette che risulta globalmente più veloce nonostante i maggiori passaggi dovuti alle diverse etichette riferenti ad uno stesso cluster.

Come si può osservare dalla figura 2.7 e 2.8, l'algoritmo "Etichetta" con $L=199$ e $p=1$ impiega 3000ms, contro i 120ms dell'algoritmo "HK" a pari parametri.

Riportiamo di seguito i tempi necessari per l'esecuzione del test eseguito su CPU Intel I5 760 da 2.8GHZ





Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
test_Confronto_HK_ET	1	362.010 s	0.511 s	
Alg_Etichetta_MEL	1628	332.844 s	49.656 s	
Vis_Clus_E_MEL	973432	283.188 s	283.188 s	
Alg_HK	1628	28.322 s	28.322 s	

Figure 2.11: Tempo di esecuzione degli algoritmi durante il test

Chapter 3

Approfondimenti

3.1 Gestione dei bordi del reticolo nell'Algoritmo Etichetta

Durante lo sviluppo del codice riguardante l'algoritmo "Etichetta" sono state scritte due varianti che differiscono nella gestione del labeling sui bordi del reticolo.

La prima variante (Alg_Etichetta_BER.m), durante il calcolo degli indici dei vicini adiacenti ad un sito, è costretta a controllare se questo sito è situato o meno su un bordo.

```
.....
% indici elemento sopra il sito centrale
% quando l'elemento centrale sulla prima riga, evitiamo che venga
% preso come indice di riga il valore 0 (fuori dal range)
rig = max([Coda_Cluster(i,1)-1, 1]);    <----- controllo(max)
col = Coda_Cluster(i,2);
if (Reticolo_AE(rig,col) == -1)
if (Reticolo_Col(rig,col) == 1)
% accodiamo a Coda_Cluster gli indici del sito visitato
Coda_Cluster = [Coda_Cluster; rig, col];
% Identifichiamo il sito con il numero del cluster di
% appartenenza
Reticolo_AE(rig,col) = Num_Clus;
else
Reticolo_AE(rig,col) = 0;
end
end
.....
```

La seconda variante (Alg_Etichetta_MEL.m, utilizzata nei test) aggiunge al reticolo una cornice di valori non significativi(zero)

```
% Metodo alternativo ottimizzante: consiste nel contornare temporaneamente
```

```

% i reticoli con siti non significativi(=0) per evitare i controlli dei siti
% adiacenti( fuori dal range) a quelli disposti sui bordi
Reticolo_Col = [zeros(1,size(Reticolo_Col,2)); Reticolo_Col;
zeros(1,size(Reticolo_Col,2))];
Reticolo_Col = [zeros(size(Reticolo_Col,1),1), Reticolo_Col, ...
zeros(size(Reticolo_Col,1),1)];
Reticolo_AE = [zeros(1,size(Reticolo_AE,2)); Reticolo_AE;
zeros(1,size(Reticolo_AE,2))];
Reticolo_AE = [zeros(size(Reticolo_AE,1),1), Reticolo_AE, ...
zeros(size(Reticolo_AE,1),1)];

```

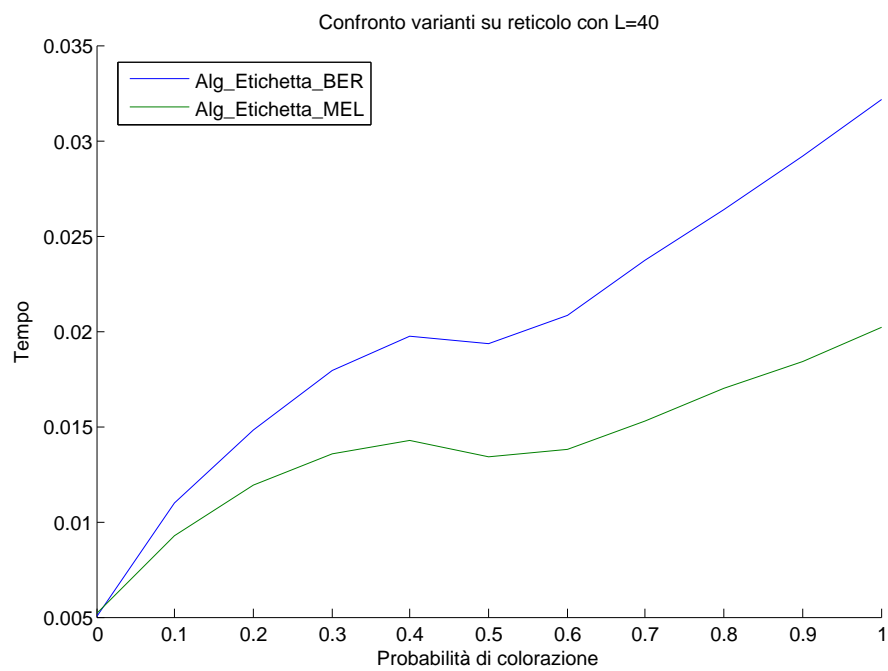
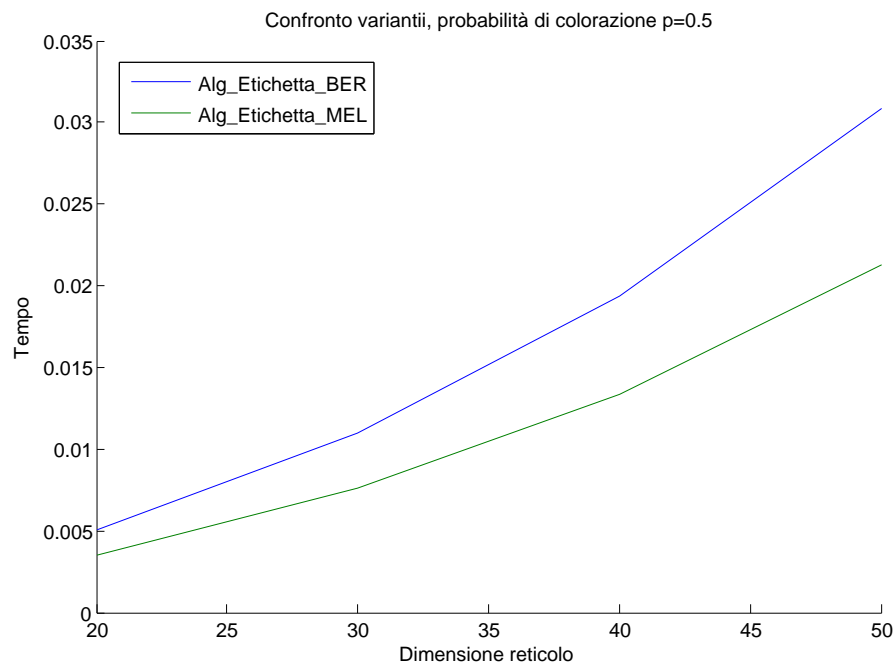
Questa aggiunta permette di non doversi preoccupare dei bordi originali del reticolo in quanto non più situati in una posizione critica; l'algoritmo ignora inoltre i nuovi bordi inseriti perchè non contenenti informazioni utili.

```

.....
% indici elemento sopra il sito centrale
rig = Coda_Cluster(i,1)-1; <----- nessun controllo
col = Coda_Cluster(i,2);
if (Reticolo_AE(rig,col) == -1)
    if (Reticolo_Col(rig,col) == 1)
        % accodiamo a Coda_Cluster gli indici del sito visitato
        Coda_Cluster = [Coda_Cluster; rig, col];
        % Identifichiamo il sito con il numero del cluster di
        % appartenenza
        Reticolo_AE(rig,col) = Num_Clus;
    else
        Reticolo_AE(rig,col) = 0;
    end
end
.....

```

Di seguito riportato un confronto sui tempi di esecuzione:



3.2 Ottimizzazione del algoritmo HK tramite calcolo parallelo

3.2.1 Descrizione e approccio intrapreso

In questa sezione verrà analizzata una variante dell'algoritmo HK pensata per poter eseguire parte dei calcoli in parallelo.

La parallelizzazione del algoritmo consiste nel suddividere il reticolo in più sottoreticoli (uno per ogni unità di calcolo). Su ogni parta viene poi applicato il classico algoritmo HK visto in precedenza; avendo quindi sottoreticoli distinti verranno utilizzate etichette (label) tali da non avere delle dipendenze: un'etichetta non appare in più di un sottoreticolo.

Finita la fase di labeling, rivisitando i bordi adiacenti dei sottoreticoli, vengono aggiornate le dimensioni e le relazioni tra i cluster trovati, rendendolo quindi un reticolo atomico.

3.2.2 Codice MATLAB

```
% Alg_HK_Par.m
function [Reticolo_AHK, Dim_Clus] = Alg_HK_Par(Reticolo)
% array contenente le dimensioni dei cluster trovati
Dim_Clus = [];
% Dimensione del reticolo da visitare
[r,c] = size(Reticolo);
% matrice delle etichette dei siti del reticolo
Reticolo_AHK= zeros(r,c);

poolobj = gcp(); % controllo che il parallel pool sia avviato
if isempty(poolobj)
    % Numero di Worker in parallelo
    num_worker = 0;
else
    % Numero di Worker in parallelo
    num_worker = poolobj.NumWorkers;
end

% Stima maggiorata del numero di cluster possibile per ogni
% sottomatrice passata ad un worker
max_clus_per_subret = ceil(1+(r*c/2)/num_worker);
max_clus = ceil(1+(r*c/2));
% lista delle relazioni tra cluster adiacenti
DimRef_Label(1:num_worker) = new_array(max_clus);
% array completo delle dimensioni dei cluster e delle relazioni
DimRef = zeros(max_clus,1);

% etichetta di partenza per ogni worker
Num_Clus = [];
for i=1:num_worker
    Num_Clus(i)= 0 + (i-1)*(max_clus_per_subret-1);
```



```

end
% numero di colonne che ogni worker deve visitare ed etichettare
column_per_worker = ceil(c/num_worker);
% sottomatrici contenenti le etichette (dopo la visita)
matrix_hk(1:num_worker) = empty_matrix();
% colonne di inizio e fine da visitare per ogni worker
c_start(1:num_worker) = 0;
c_end(1:num_worker) = 0;

% iniziamo la visita in parallelo sulle sottomatrici
parfor worker_id=1:num_worker
% quantita di worker che lavorano ad un numero
% pieno(column_per_worker) di colonne
n_full = c - num_worker*(column_per_worker-1);
% calcoliamo colonna di inizio e colonna finale da visitare
% all'interno del reticolo
if worker_id <= n_full
    c_start(worker_id) = 1 + column_per_worker*(worker_id-1);
    c_end(worker_id) = c_start(worker_id) + column_per_worker - 1;
else
    c_start(worker_id)=1+n_full*column_per_worker+ ...
        ... (column_per_worker-1)*(worker_id-n_full-1);
    c_end(worker_id) = c_start(worker_id) + column_per_worker - 2;
end
% copia locale della parte interessata del reticolo
subret = Reticolo(:,c_start(worker_id):c_end(worker_id));
subret_hk = zeros(size(subret));

for col = 1: c_end(worker_id)-c_start(worker_id)+1
for rig = 1:r

% se subret    occupato/colorato
if (subret(rig,col) == 1)
    % verifichiamo in che situazione ci troviamo tenendo conto
    % anche della situazione agli estremi
    Situaz = 0;
    % se non siamo sulla prima colonna controlliamo solo il vicino
    % sinistro
    if col > 1
        % Etichette degli elemento adiacenti sinistro
        Label_Sx = subret_hk(rig,col-1);
        Situaz = sign(Label_Sx);
    end
    % se non siamo sulla prima riga controlliamo solo il vicino
    % superiore
    if rig > 1
        % Etichette degli elemento adiacenti superiore
        Label_Up = subret_hk(rig-1,col);
        Situaz = Situaz + 2*sign(Label_Up);
    end
    switch Situaz
        % Situaz=0 -> vicini superiore e sinistro non colorati
        case 0
            % nuovo cluster. Aumentiamo il numero di cluster

```

```

    % trovati nel reticolo
    Num_Clus(worker_id) = Num_Clus(worker_id)+1;
    subret_hk(rig,col) = Num_Clus(worker_id);
    DimRef_Label(worker_id) = add_in_array(DimRef_Label(worker_id),Num_Clus(worker_id));
% Situaz=1 -> vicini sinistro colorato
case 1
    ref_ok = Label_Sx;
    while get_in_array(DimRef_Label(worker_id),ref_ok) < 0
        ref_ok = -get_in_array(DimRef_Label(worker_id),ref_ok)
    end
    % incrementiamo il contatore del cluster a cui fa
    % riferimento il vicino
    DimRef_Label(worker_id) = add_in_array(DimRef_Label(worker_id),ref_ok);
    % etichettiamo direttamente col riferimento
    subret_hk(rig,col) = ref_ok;
% Situaz=2 -> vicini superiore colorato
case 2
    ref_ok = Label_Up;
    while get_in_array(DimRef_Label(worker_id),ref_ok) < 0
        ref_ok = -get_in_array(DimRef_Label(worker_id),ref_ok)
    end
    % incrementiamo il contatore del cluster a cui fa
    % riferimento il vicino
    DimRef_Label(worker_id) = add_in_array(DimRef_Label(worker_id),ref_ok);
    % etichettiamo direttamente col riferimento
    subret_hk(rig,col) = ref_ok;
% Situaz=3 -> vicini superiore e sinistro colorati
case 3
    % cerchiamo ricorsivamente l'etichetta dei cluster
    % a cui fanno riferimento i vicini
    ref_up = Label_Up;
    ref_sx = Label_Sx;
    while get_in_array(DimRef_Label(worker_id),ref_up) < 0
        ref_up = -get_in_array(DimRef_Label(worker_id),ref_up)
    end
    while get_in_array(DimRef_Label(worker_id),ref_sx) < 0
        ref_sx = -get_in_array(DimRef_Label(worker_id),ref_sx)
    end
    % valutazioni sulle etichette trovate nei while
    if ref_sx < ref_up % ref SX < ref UP
        % coloriamo il sito con l'etichetta minore (sx)
        subret_hk(rig,col) = ref_sx;
        % incrementiamo il contatore del cluster della
        % etichetta minore con il valore del contatore
        % del cluster dell'etichetta maggiore
        DimRef_Label(worker_id) = sum_in_array(DimRef_Label(worker_id), ...
            ... ref_sx, get_in_array(DimRef_Label(worker_id),ref_up)+1);
        % il contatore dei cluster diventer quello del
        % cluster con etichetta minore tra i due
        DimRef_Label(worker_id) = set_in_array(DimRef_Label(worker_id), ...
            ... ref_up, -ref_sx);
    elseif ref_sx > ref_up % ref SX > ref UP
        subret_hk(rig,col) = ref_up;
        DimRef_Label(worker_id) = sum_in_array(DimRef_Label(worker_id), ...

```

```

        ... ref.up, get_in_array(DimRef_Label(worker_id), ref_sx)+1);
DimRef_Label(worker_id) = set_in_array(DimRef_Label(worker_id), ...
        ... ref_sx, -ref.up)
elseif ref_sx == ref.up % SX = UP
    % coloriamo il sito con l'etichetta del cluster
    % di riferimento
    subret_hk(rig,col) = ref.up;
    % incrementiamo il contatore del cluster
    % di riferimento
    DimRef_Label(worker_id) = add_in_array(DimRef_Label(worker_id), ref.up);
end
end
end
end
matrix_hk(worker_id) = matrix(subret_hk);
end

% ricostruiamo il reticolo delle etichette completo e l'array delle
% dimensioni e dei riferimenti tra cluster
for worker_id=1:num.worker
    DimRef = DimRef + get_array(DimRef_Label(worker_id));
    Reticolo_AHK(:,c_start(worker_id):c_end(worker_id)) = get_matrix(matrix_hk(worker_id));
end

% controllo bordi tra sub-reticoli per aggiornare i riferimenti
for worker_id=num.worker-1:-1:1
    for rig=1:r
        % controlliamo solo per i worker che hanno lavorato su almeno
        % una colonna
        if c_end(worker_id) < c
            ref_sx = Reticolo_AHK(rig,c_end(worker_id));
            ref_dx = Reticolo_AHK(rig,c_end(worker_id)+1);
            % se entrambi i siti sono etichettati
            if ref_sx>0 && ref_dx>0
                while DimRef(ref_sx) < 0
                    ref_sx = -DimRef(ref_sx);
                end
                while DimRef(ref_dx) < 0
                    ref_dx = -DimRef(ref_dx);
                end
                if ref_sx ~= ref_dx
                    DimRef(ref_sx) = DimRef(ref_sx) + DimRef(ref_dx);
                    DimRef(ref_dx) = -ref_sx;
                end
            end
        end
    end
end

% Manteniamo solo le dimensioni del vettore DimRef_Label (eliminando i riferimenti)
Dim_Clus = DimRef;
Dim_Clus(DimRef < 1) = [];
end

```

3.2.3 Risultati e osservazioni

Nel grafico di figura 3.1 vengono confrontati l'algoritmo HK e la variante parallelizzata su un reticolo di $L=101$, inoltre abbiamo rappresentato l'algoritmo HK al variare del numero di worker, ovvero all'aumentare delle unità elaboranti ogni singolo sottoreticolo opportunamente suddiviso.

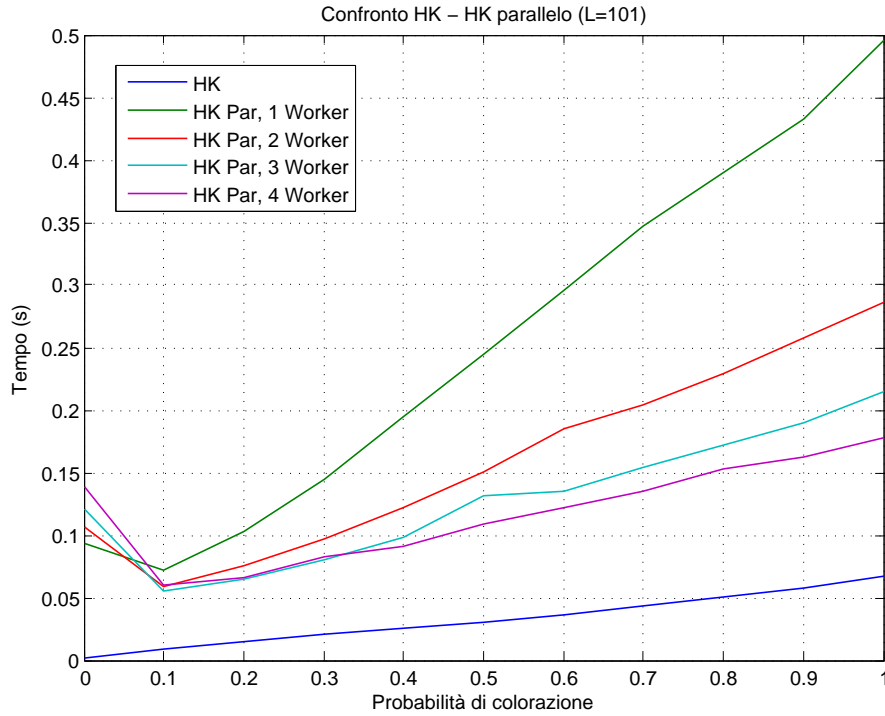


Figure 3.1: Confronto HK - HK parallelo

Possiamo infine concludere dai risultati inaspettati che la parallelizzazione dell'algoritmo risulti più lenta della versione sequenziale, questa anomalia è probabilmente dovuta ad una gestione non efficiente delle risorse utilizzate dalle funzioni parallelizzanti di MATLAB. Non avendo malgrado ottenuto vantaggi dall'utilizzo della parallelizzazione, durante lo svolgimento delle prove riguardanti la relazione è stata utilizzata la versione sequenziale.

3.3 Percolazione su reticolo 3D

3.3.1 Descrizione e approccio intrapreso

Come già detto nel cenno teorico idealmente vorremo studiare sistemi di dimensioni infinite ma non essendo possibile ci limitiamo a sistemi chiusi

rappresentati da reticoli in d -dimensioni; negli esperimenti precedenti abbiamo considerato reticoli con due dimensioni(2D) e quindi rappresentabili da una matrice, in questo approfondimento ci proponiamo di verificare sperimentalmente la soglia di percolazione nel caso di tre dimensioni(3D) ed effettuare valutazioni e confronti con i dati precedentemente raccolti dalle simulazioni a 2D.

I reticoli presi in considerazione si estendono quindi su 3 assi (X,Y,Z) e di conseguenza anche i cluster ad esso appartenenti; un cluster è detto percolante quando si estende (sull'asse Z) da una faccia del reticolo a quella opposta.

3.3.2 Codice MATLAB

```
function [Reticolo_Col] = CreaCol.Ret3D (dim, Prob_Col)
    % Creazione e colorazione del reticolo con lato di dimensione "dim" con
    % probabilit di colorazione "Prob_Col"
    Reticolo_Col = rand(dim,dim,dim) < Prob_Col;
end
```

```
function [Reticolo_AE, Dim_Clus] = Alg_Etichetta_3D (Reticolo_Col)
    % Algoritmo Etichetta : Ricerca e numerazione dei cluster in un
    % reticolo colorato, cio associa ad ogni sito il cluster di appartenenza.
    % vettore contenente le dimensioni dei cluster trovati
    Dim_Clus = 0;
    % numero dei cluster trovati
    Num_Clus=0;
    % Matrice delle stesse dimensioni del reticolo colorato "Reticolo_Col", in
    % cui in ogni sito salviamo l'etichetta identificativa del cluster di appartenenza
    Reticolo_AE=zeros(size(Reticolo_Col)+2)-1;
    % contorno temporaneo reticoli con siti non significativi(=0) per evitare
    % i controlli dei siti adiacenti( fuori dal range) a quelli disposti sui bordi
    dimension = size(Reticolo_AE);
    tmp = zeros(dimension);
    tmp(2:dimension(1)-1,2:dimension(2)-1,2:dimension(3)-1) = Reticolo_Col;
    Reticolo_Col = tmp;

    % per ogni colonna e livello di profondita del Reticolo_Col
    for z = 2:size(Reticolo_Col,3)-1
        for y = 2:size(Reticolo_Col,2)-1
            % per ogni riga(per ogni elemento del vett col) del Reticolo_col
            for x = 2:size(Reticolo_Col,1)-1
                % se Reticolo_Col(x,y,z) occupato e non visitato aggiungiamo alla coda
                % e visitiamo il cluster di appartenenza
                if (Reticolo_AE(x,y,z) == -1)
                    if (Reticolo_Col(x,y,z) == 1)
                        Num_Clus=Num_Clus+1;
                        Coda_Cluster = [x y z];
```

```

Reticolo_AE(x,y,z) = Num_Clus;
% visita al cluster
[Reticolo_AE,Coda_Cluster] = Vis_Clus_E3D(Reticolo_Col, ...
    ... Reticolo_AE,Coda_Cluster,Num_Clus);
% Salviamo la dimensione dell'ultimo cluster trovato
Dim_Clus(Num_Clus+1) = size(Coda_Cluster,1);
%finito di visitare il cluster si procede con gli altri siti
else
    Reticolo_AE(x,y,z) = 0;
end
end
end
end
end
% rimozione dei siti non significativi(=0)
Reticolo_Col = Reticolo_Col(2:size(Reticolo_Col,1)-1,2:size(Reticolo_Col,2)-1, ...
    ... 2:size(Reticolo_Col,3)-1);
Reticolo_AE = Reticolo_AE(2:size(Reticolo_AE,1)-1,2:size(Reticolo_AE,2)-1, ...
    ... 2:size(Reticolo_AE,3)-1);
Dim_Clus(1) = [];
end

```

```

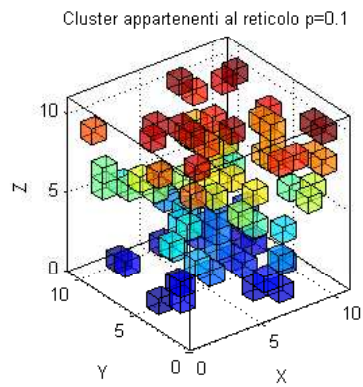
function [Reticolo_AE,Coda_Cluster] = Vis_Clus_E3D(Reticolo_Col, Reticolo_AE, Coda_Clus

% valori utilizzati per calcolare gli indici dei siti adiacenti da visitare
face_index = [-1  0  0 ;
               1  0  0 ;
               0 -1  0 ;
               0  1  0 ;
               0  0 -1 ;
               0  0  1 ];

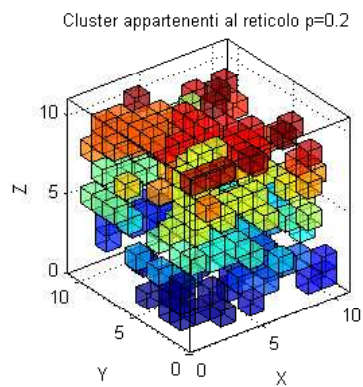
i=0;
while i < size(Coda_Cluster,1)
    i = i+1;
    % controllo dei siti adiacenti (sopra,destra...)
    for a=1:6 % ciclo per controllare i siti adiacenti su ogni faccia
        x = Coda_Cluster(i,1) + face_index(a,1);
        y = Coda_Cluster(i,2) + face_index(a,2);
        z = Coda_Cluster(i,3) + face_index(a,3);
        if (Reticolo_AE(x,y,z) == -1)
            if (Reticolo_Col(x,y,z) == 1)
                % accodiamo a Coda_Cluster gli indici del sito visitato
                Coda_Cluster = [Coda_Cluster; x,y,z];
                % Identifichiamo il sito con il numero del cluster di
                % appartenenza
                Reticolo_AE(x,y,z) = Num_Clus;
            else
                Reticolo_AE(x,y,z) = 0;
            end
        end
    end
end
end
end

```

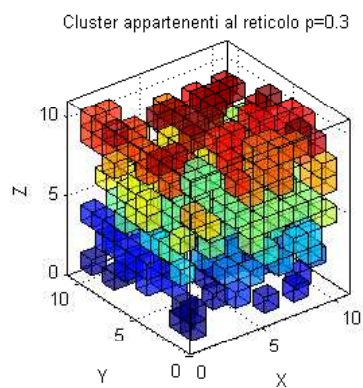
end



PERCOLAZIONE
NON PRESENTE



PERCOLAZIONE
NON PRESENTE



PERCOLAZIONE
NON PRESENTE

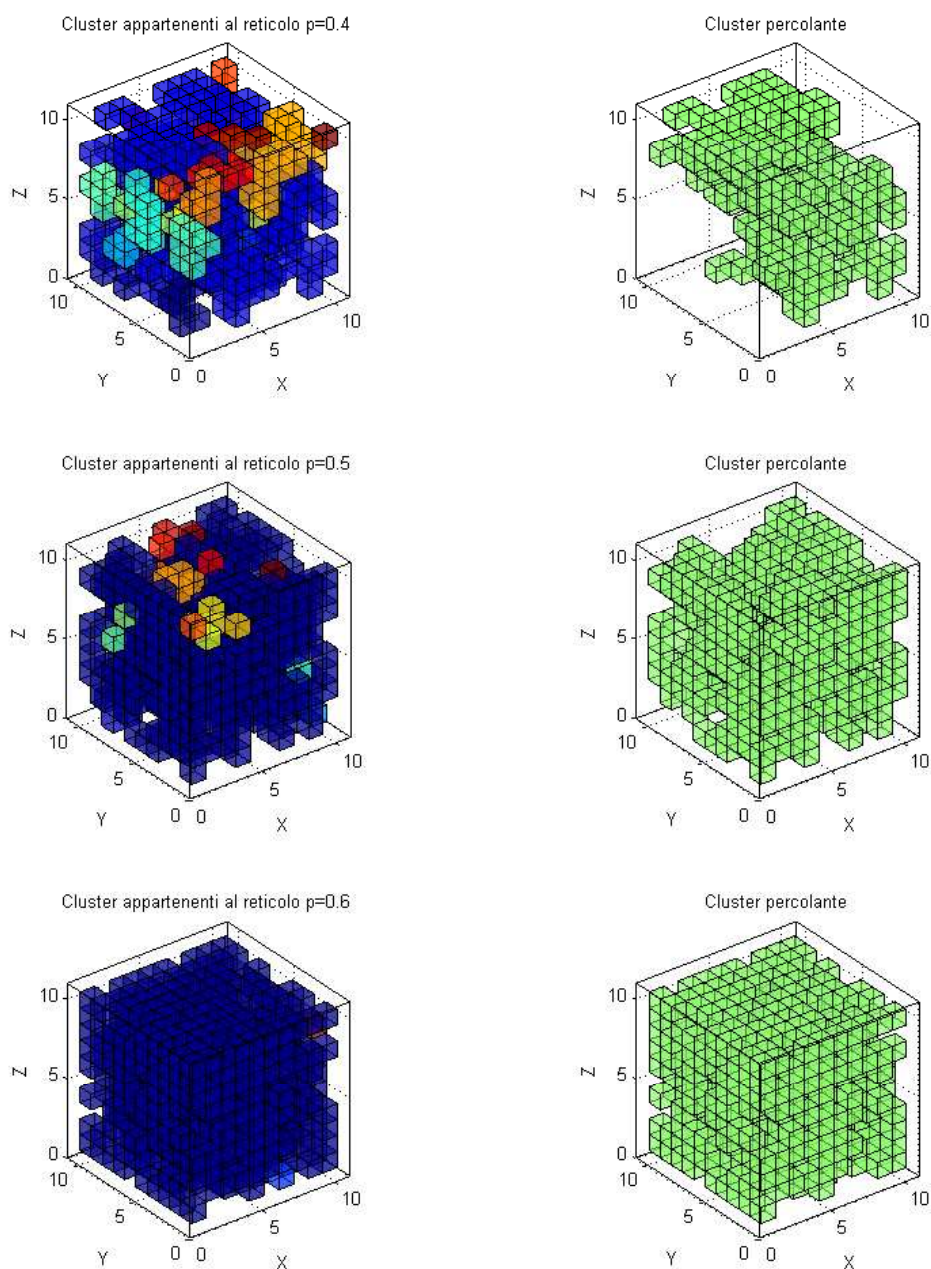


Figure 3.2: Percolazione su reticoli 3D

3.3.3 Frequenza di percolazione su reticolo 3D

Di seguito analizzeremo la frequenza di percolazione, in base alla probabilità di colorazione dei siti, su un reticolo 3D; la prova viene svolta su reticoli di diverse dimensioni.

Ricordiamo che il reticolo è definito dalla grandezza del lato L e di conseguenza possiede L^3 siti.

3.3.4 Codice MATLAB frequenza di percolazione 3D

```
% Test sulla frequenza di percolazione su un reticolo di 3-dimensioni
clear
clc
% numero probabilit
np=30;
% probabilita nel range [0.05 0.95]
np_range=linspace(0.05,0.95,np);
% numero di tentativi per ogni probabilit
tent = 400;

% dimensione reticolo
for dim=[10,20,30]
    Freq_Perc = zeros(1,np);
    % numero di tentativi parallelizzati
    parfor i=1:np
        % probabilit di colorazione
        p = np_range(i);
        perc = zeros(1,tent);
        % i casi limite con 0 e 1 sappiamo che descrivono rispettivamente
        % l'assenza di percolazione e la presenza certa
        for t = 1 : tent
            Reticolo=CreaColRet3D(dim,p);
            [Reticolo_AE, Dim_Clus] = Alg_Etichetta_3D(Reticolo);
            if Ricer_Percol3D(Reticolo_AE) ~= 0
                Freq_Perc(i) = Freq_Perc(i) + 1;
                perc(t) = 1;
            end
        end
        % CALCOLO FREQUENZA DI PERCOLAZIONE (vPerc)
        Freq_Perc(i) = Freq_Perc(i)/tent;
        % CALCOLO DELL'ERRORE (associato ad ogni probabilit p)
        % mediante funzione matlab 'std' che ritorna la deviazione standard
        Err_Freq_Perc(i) = std(perc)/sqrt(tent);
    end
end
% GRAFICO
hold all
errorbar(np_range,Freq_Perc,Err_Freq_Perc);
end
```

3.3.5 Osservazioni e Differenze tra percolazione 2D e 3D

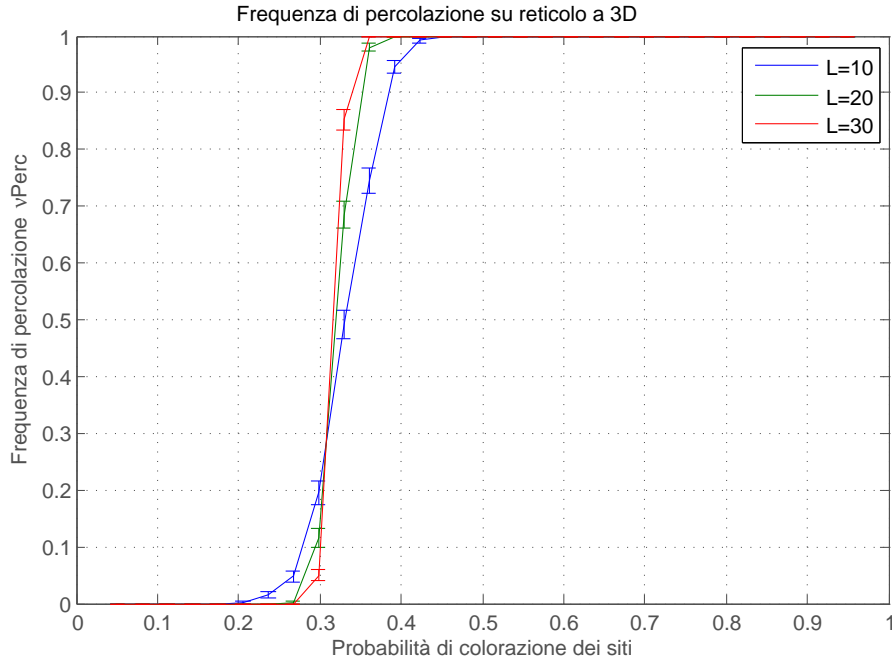


Figure 3.3: Frequenza di percolazione su reticolo 3D

Da ciò che viene illustrato nel grafico di figura 3.3 possiamo concludere che come precedentemente affermato per i reticoli 2D, all'aumentare della dimensione del reticolo, la funzione risulti sempre più rapida nel passaggio dalla probabilità di colorazione con assenza certa di percolazione alla probabilità dove vi è presenza certa.

Diversamente da come visto in precedenza, questo passaggio si assesta in un intorno del valore di probabilità $p=0.3$ che risulta essere il valore della soglia di percolazione per reticoli 3D. Questa minor soglia di probabilità è dovuta alla presenza di più siti adiacenti, per la precisione con 3 dimensioni si hanno 6 siti adiacenti (assi X,Y,Z) invece che 4 (assi X,Y). Quindi si ha una maggiore possibilità di "espansione" di un cluster.