Studio della Percolazione tramite Metodi di Simulazione Numerica

Daniele Molinari

6/02/2025

Abstract

Questa analisi si propone di studiare il fenomeno della **percolazione** su un reticolo quadrato bidimensionale. Per raggiungere tale obiettivo, è stato implementato l'algoritmo di cluster finding di **Hoshen-Kopelman** e confrontato con l'algoritmo sviluppato a lezione, validandone le performance e i risultati. L'analisi si conclude con uno studio dettagliato dei risultati, con particolare attenzione alla soglia di percolazione.

1 Introduzione

La **percolazione** è un fenomeno fisico e matematico che descrive il comportamento di sistemi complessi composti da unità interconnesse. Nonostante il problema sia nato nello studio del trasporto di fluidi attraverso materiali porosi, il modello di percolazione si applica a una vasta gamma di fenomeni naturali e artificiali, come la propagazione di epidemie, la diffusione di incendi nelle foreste, il percolato della spazzatura e persino la preparazione del caffè.

1.1 Reticolo

Nel contesto di questo elaborato consideriamo la percolazione su un **reticolo quadrato bidimensionale**. Un reticolo è definito come un insieme ordinato di punti o **siti**, in cui ogni sito è connesso ai suoi primi vicini (a sinistra, destra, sopra e sotto) e ai secondi vicini (elementi sulla diagonale di un sito). Questa struttura locale è replicata in modo uniforme, a meno dei siti di bordo [1].

[1] I siti situati ai bordi del reticolo presentano criticità nella loro gestione, poiché richiedono un trattamento differenziato. Questo aspetto è emerso chiaramente durante lo sviluppo dell'algoritmo di cluster finding analizzato a lezione.

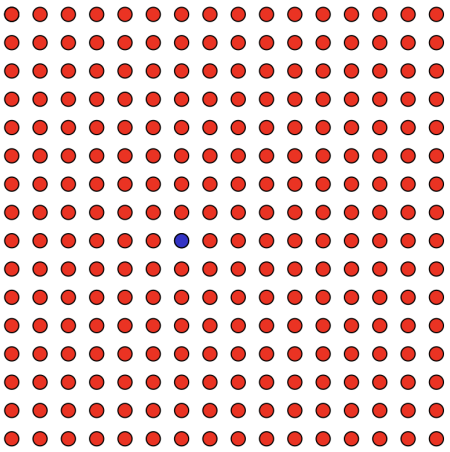


Figure 1: Reticolo quadrato

L'aspetto centrale del modello di percolazione è l'assegnazione casuale di uno stato (colorato o non colorato) ai siti del reticolo, secondo una probabilità uniforme **p**. Questo genera un reticolo partito in due sottoinsiemi: i siti colorati e quelli non colorati. I siti colorati, a loro volta, possono essere raggruppati in **cluster**, definiti come insiemi di siti connessi attraverso cammini composti da primi vicini colorati. Nella Figura 2 è possibile vedere un reticolo con una serie di cluster.

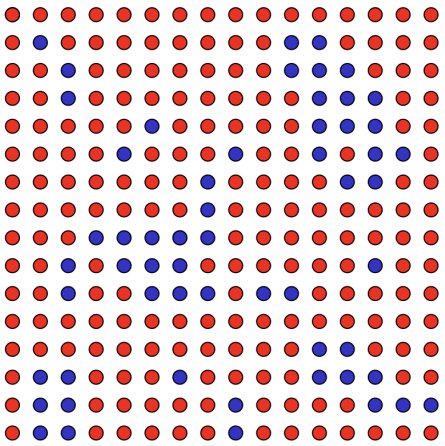


Figure 2: Reticolo con formazione di cluster

1.2 Fenomeno della Percolazione

In questo elaborato si intende analizzare il fenomeno della percolazione, che si verifica quando un cluster collega due lati opposti di un reticolo, come illustrato nella Figura 3. Nel limite termodinamico, ovvero in un reticolo di dimensioni infinite, questa transizione tra assenza e presenza di percolazione si configura come una proprietà emergente del sistema. Infatti, tale fenomeno si manifesta solo se la probabilità **p** supera una soglia critica p_c. Se **p** è inferiore a p_c, la probabilità di percolazione è nulla, mentre se **p** è superiore, essa diventa pari a **1**. Nei prossimi capitoli, verranno approfonditi e verificati i dettagli relativi a questa soglia critica.

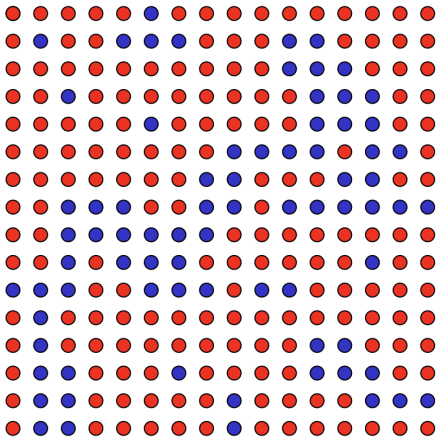


Figure 3: Cluster percolante tra i lati SX e DX

2 Algoritmo di Hoshen-Kopelman

L'algoritmo di Hoshen-Kopelman è un algoritmo di **cluster finding** che permette di identificare i cluster all'interno di un reticolo. Il suo funzionamento è intuitivo e si sviluppa nel seguente modo: il reticolo viene esplorato sito per sito, procedendo per colonne, partendo dall'angolo in alto a sinistra e arrivando all'angolo in basso a destra. Quando si incontra un sito, si verifica se è colorato. Se il sito è colorato, si controllano i primi vicini **sopra** e a **sinistra** e assegnamo una label che identificherà il cluster di appartenenza (il primo cluster sarà contrassegnato con il valore 1). Esistono tre possibili casi da considerare durante l'assegnazione di una label:

- **Nuovo cluster**: Se il sito occupato non è connesso ad altri siti occupati sopra e a sinistra allora inizia un nuovo cluster (e.g., in pratica andiamo ad incrementare di uno la variabile che denota la label dei cluster);
- **Connessione a cluster esistente**: Se il sito è occupato e c'é un primo vicino sopra o a sinistra occupato (uno solo dei due) il sito in esame prende il valore del sito primo vicino occupato, analogamente se i suoi primi vicini sono entrambi occupati ma con lo stesso label.
- **Collisione tra cluster**: Se il sito è occupato e i suoi primi vicini (sopra e a sinistra) sono occupati ma con etichette diverse, il sito viene etichettato con il valore **minore** tra le due etichette. Per gestire il fatto che i cluster con etichette diverse appartengono in realtà allo stesso cluster, si utilizza una tecnica di **multilabelling**. Questa tecnica sfrutta un array ausiliario, che mappa la label più **grande** incontrata con quella più piccola, permettendo di unificare correttamente i cluster durante l'analisi.

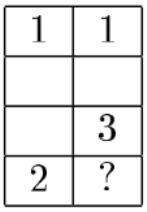


Figure 4: Elementi dello stesso cluster ma con label diversa

Esempio. Prendendo come riferimento la Figura 4, nel caso di una collisione assegneremo al sito contrassegnato con il valore del cluster "3" il valore "2" nell'array ausiliario. Questo permette di tracciare il vero cluster di appartenenza tramite una rete di "puntatori", in cui il valore "3" viene risolto come appartenente al cluster identificato da "2" (ad esempio: $3 \to 2$). Tale approccio consente di determinare in modo efficiente il cluster finale durante l'analisi.

2.1 Implementazione dell'algoritmo

L'algoritmo è stato implementato utilizzando il linguaggio MATLAB ed è strutturato in tre componenti principali:

- **Funzione principale**: Si occupa dell'inizializzazione delle strutture dati fondamentali, tra cui la matrice di input, la matrice delle etichette e il vettore `labelOfLabel`, utilizzato per tenere traccia del *rootCluster*[2]. La funzione gestisce i tre casi principali basandosi sui valori dei primi vicini (sopra e a sinistra), aggiornando opportunamente le etichette e determinando, infine, se si verifica la percolazione. Di seguito il ciclo principale che gestisce gli elementi di frontiera e i tre casi principali.

```
%...

for i = 1:length(valid)
    sito = valid(i);

    % Elemento sinistro
    if(sito <= L)
        left = 0;
    else
        left = matrix.label(sito - L);
    end


    % Elemento superiore
    if(mod(sito, L) == 1)
        top = 0;
    else
        top = matrix.label(sito - 1);
    end
```

[2]Il cluster radice minimo a cui ciascun cluster appartiene

```
        % Caso 1
        if(left == 0 && top == 0)
            matrix.label(sito) = cluster;
            cluster = cluster + 1;

        % Caso 2
        elseif((left == 0 && top ~= 0) || (left ~= 0 &&
           top == 0) || (left == top))
            if left > top
                matrix.label(sito) = Find(left);
            else
                matrix.label(sito) = Find(top);
            end

        % Caso 3
        elseif(left ~= top && left ~= 0 && top ~= 0)
            Union(left,top)
            matrix.label(sito) = Find(left);
        end
    end
    % ...
```

- **Funzione Find**: Gli indici del vettore `labelOfLabel` rappresentano i cluster, mentre i valori associati corrispondono al rispettivo *rootCluster*, ossia il cluster radice minimo di appartenenza. La funzione `Find` calcola il *rootCluster* seguendo la catena di dipendenze nel vettore `labelOfLabel`. Durante il processo, effettua un'operazione di compressione del percorso (*path compression*) aggiornando direttamente i valori dei cluster incontrati, riducendo così la lunghezza delle catene future e migliorando l'efficienza complessiva dell'algoritmo.

```
    function out = Find(x)
        out = x;

        % Trovo la radice
        while matrix.labelOfLabel(out) ~= out
            out = matrix.labelOfLabel(out);
        end

        % Aggiorno tutto il percorso al valore minimo
        while matrix.labelOfLabel(x) ~= x
            tmp = matrix.labelOfLabel(x);
            matrix.labelOfLabel(x) = out;
            x = tmp;
        end
    end
```

- **Funzione Union**: La funzione `Union` viene utilizzata per gestire il caso in cui i primi vicini sopra e a sinistra di un sito appartengano a cluster distinti (vedi Caso 3). Il suo compito è unire i due cluster, assegnando al *rootCluster* del cluster con valore maggiore quello del *rootCluster* più piccolo, garantendo così una rappresentazione coerente e compatta delle relazioni tra i cluster.

```
function Union(left, top)
    rootLeft = Find(left);
    rootTop = Find(top);

    if rootLeft > rootTop
        matrix.labelOfLabel(rootLeft) = rootTop; %
           Unisce left a top
    else
        matrix.labelOfLabel(rootTop) = rootLeft; %
           Unisce top a left
    end
end
```

2.2 Identificazione percolazione

Per determinare la percolazione dopo l'esecuzione dell'algoritmo di Hoshen-Kopelman, vengono individuati i cluster presenti ai bordi opposti della griglia. Successivamente, per ciascun cluster si identifica il rispettivo rootCluster, così da verificare se esiste almeno un cluster che collega entrambi i bordi che condividono lo stesso rootCluster. Di seguito è riportata l'implementazione utilizzata per il caso Top-Bottom:

```
% ...

% Identifico i cluster al bordo Top
top = unique(matrix.label(1:L:L*(L-1) + 1));
top = top(top > 0);
realTop = zeros(length(top), 1);

% Identifico il rootCluster di appartenenza
for i = 1:length(top)
    realTop(i) = Find(top(i));
end

realTop = unique(realTop);

% Identifico i cluster al bordo Bottom
bottom = unique(matrix.label(L:L:L*L));
bottom = bottom(bottom > 0);
realBottom = zeros(length(bottom), 1);

% Identifico il rootCluster di appartenenza
for i = 1:length(bottom)
    realBottom(i) = Find(bottom(i));
end

realBottom = unique(realBottom);

% Verifico l'intersezione tra i rootCluster
if (~isempty(intersect(realTop, realBottom)))
    matrix.percolazioneTB = 1;
end

% ...
```

2.3 Performance

La correttezza dell'algoritmo è stata verificata attraverso un benchmark, confrontando i risultati ottenuti nel calcolo della percolazione con quelli forniti dall'algoritmo analizzato in aula. Di seguito è riportato un estratto del codice utilizzato per verificare se entrambi gli algoritmi identificano la percolazione in modo coerente:

```
% Algoritmo A e B a confronto Top-Bottom
if (A.percolazioneTB ~= B.percolazioneTB)
    error("Risultato non corretto");
end

% Algoritmo A e B a confronto Left-Right
if (A.percolazioneLR ~= B.percolazioneLR)
    error("Risultato non corretto");
end
```

Performance. Per quanto concerne le performance, sono stati eseguiti due tipi di confronti:

- Confronto delle performance a **taglia fissa** variando la probabilità.

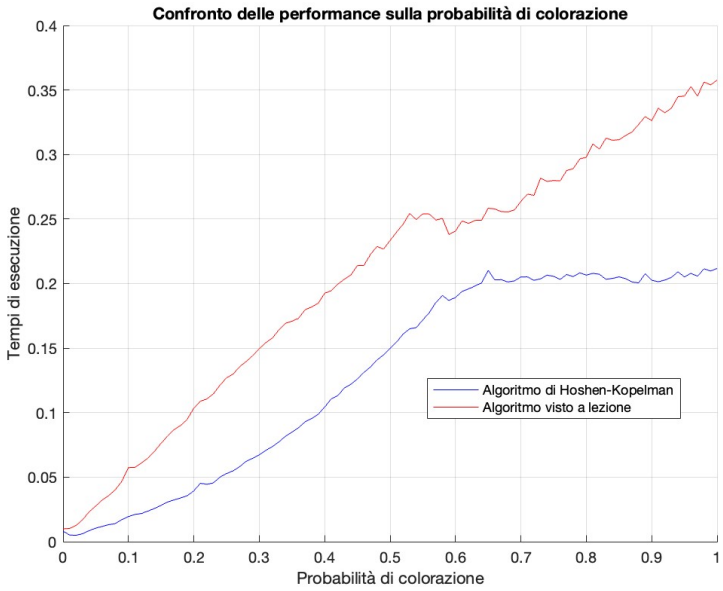


Figure 5: Taglia fissa al variare della probabilità

- Confronto delle performance a probabilità fissa e al **variare della taglia** del reticolo

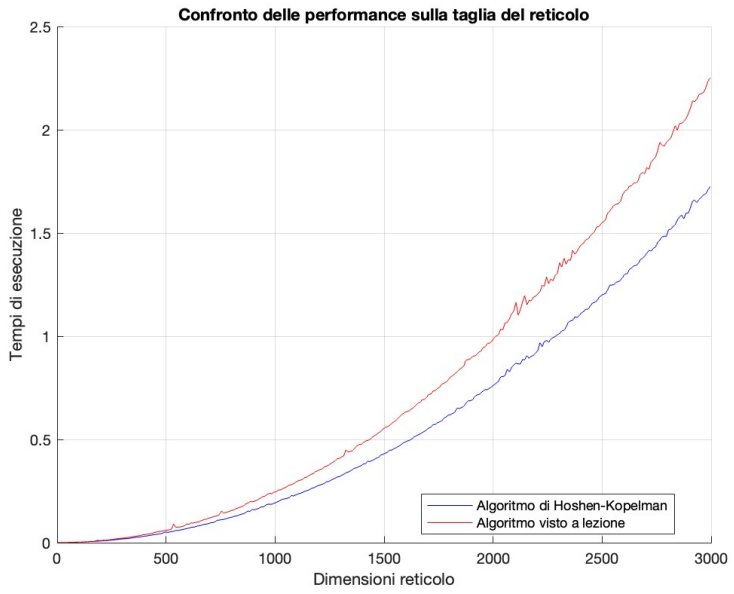


Figure 6: Taglia variabile e probabilità fissata

Per ottenere un confronto più preciso è necessario il calcolo della media dei tempi di esecuzione nelle diverse configurazioni in merito alla dimensione del problema, in modo da considerare le fluttuazioni che potrebbero influire sul risultato. Pertanto, oltre a calcolare il valore medio, è importante prendere in considerazione l'errore, definito dalla seguente formula:

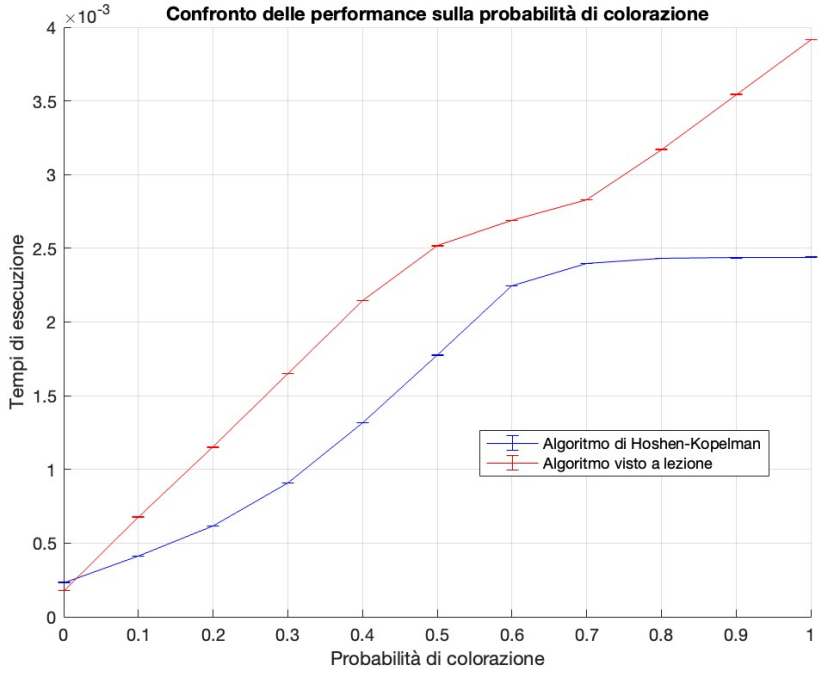
$$\tilde{m} \pm \frac{\tilde{\sigma}}{\sqrt{n}}$$

Dove $\tilde{m}$ è la media dei tempi di esecuzione, $\tilde{\sigma}$ è la deviazione standard e n è il numero di misure.

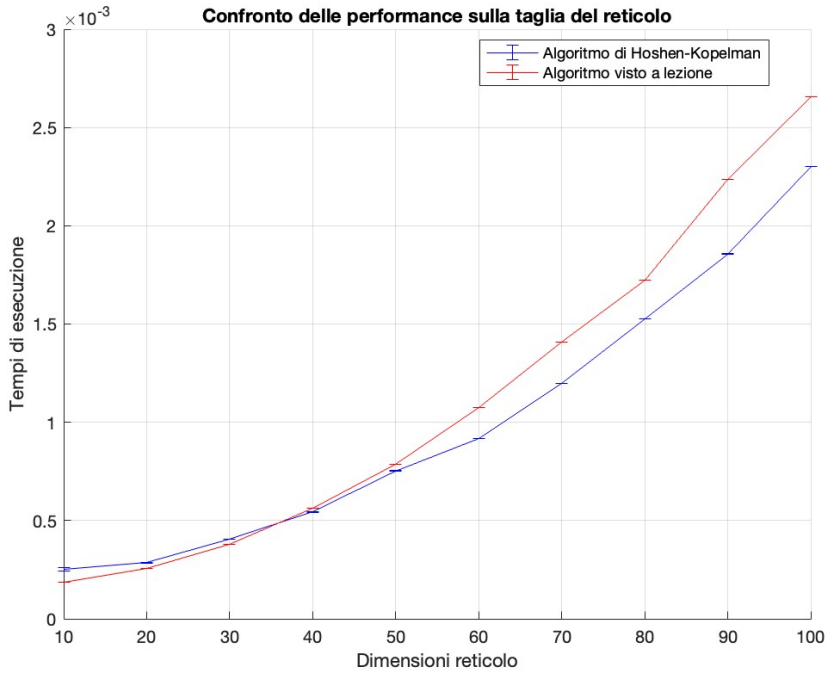
A tale scopo è stato sviluppato uno script (**confronto_performance.m**) per effettuare una serie di misurazioni, ripetute circa 1000 volte per ciascuna configurazione[3]. L'obiettivo è stato calcolare le performance medie degli algoritmi e stimare accuratamente l'errore associato.

Nelle Figure 7a e 7b sono mostrati i risultati delle performance, con i relativi errori riportati per ciascuna misura.

[3]Per configurazione si intende una combinazione fissata di probabilità e dimensione del reticolo, su cui vengono eseguite più misurazioni per calcolarne il tempo di esecuzione medio e il relativo errore.



(a) Grafico media su prob. colorazione



(b) Grafico media taglia del reticolo

Figure 7: Confronto tra le medie dei due algoritmi: probabilità di colorazione e dimensione del reticolo.

Di seguito viene riportato un estratto del codice utilizzato per il benchmark. Questo frammento calcola il tempo medio di esecuzione per diverse configurazioni della dimensione del reticolo, ripetendo ogni esperimento un numero fisso di volte (m). Inoltre, per ogni configurazione, viene calcolato anche l'errore associato alla media, stimato come deviazione standard divisa per la radice quadrata del numero di ripetizioni. Analogamente il codice per il calcolo del tempo medio al variare della probabilità di colorazione sarà simile ma il valore che varia sarà la probabilità.

```
%...

% ===== Per taglia del reticolo =====
counter = 1;
N = 100;

for i = 10:10:N
    p = 0.6;
    matrix = rand(i) < p;

    % Ripeti l'esperimento m volte per ciascuna
       configurazione di dimensione
    tempiA = zeros(m, 1);
    tempiB = zeros(m, 1);

    for j = 1:m
        tic;
        A = HK(p, i, matrix);
        tempiA(j) = toc;

        tic;
        B = CercaCluster2(i, p, matrix);
        tempiB(j) = toc;

        if A.percolazioneTB ~= B.percolazioneTB || A.
           percolazioneLR ~= B.percolazioneLR
            error("Risultato non corretto");
        end
    end

    % Calcola la media e l'errore
    tempiTagliaA(counter) = mean(tempiA);
    tempiTagliaB(counter) = mean(tempiB);
    erroreTagliaA(counter) = std(tempiA) / sqrt(m);
    erroreTagliaB(counter) = std(tempiB) / sqrt(m);

    counter = counter + 1;
end

% ...
```

3 Calcolo soglia di percolazione

In questo capitolo ci focalizzeremo sul calcolo della **soglia di percolazione**, che, al crescere indefinitamente della dimensione del reticolo, converge verso il comportamento tipico della funzione di Heaviside, come mostrato in Figura 8.

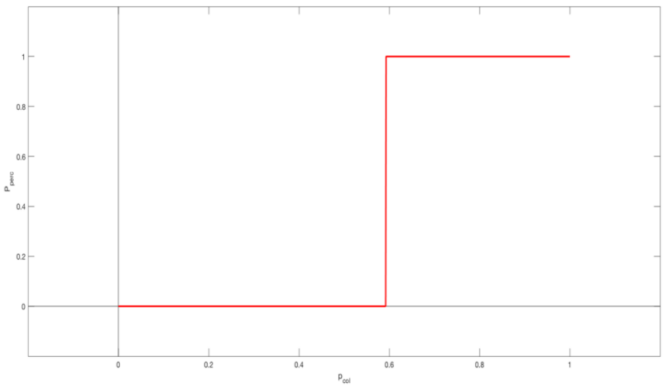


Figure 8: soglia di percolazione asintotica

A tale scopo è stato implementato uno script (**soglia_percolazione.m**) che esegue, per ciascuna configurazione, N misurazioni al fine di calcolare il valore medio della probabilità di percolazione per ogni probabilità di colorazione. Questo approccio consente di ottenere un grafico che mostra i punti corrispondenti alle probabilità di percolazione, accompagnati dai relativi errori, fornendo così una rappresentazione visiva accurata della distribuzione dei risultati ottenuti mediante la funzione **errorbar**. Nella Figura 9 è possibile vedere che il grafico tende ad una sigmoide, in quanto solo asintoticamente (e quindi al tendere della taglia del reticolo all'infinito) andrà a formarsi una funzione a gradino.

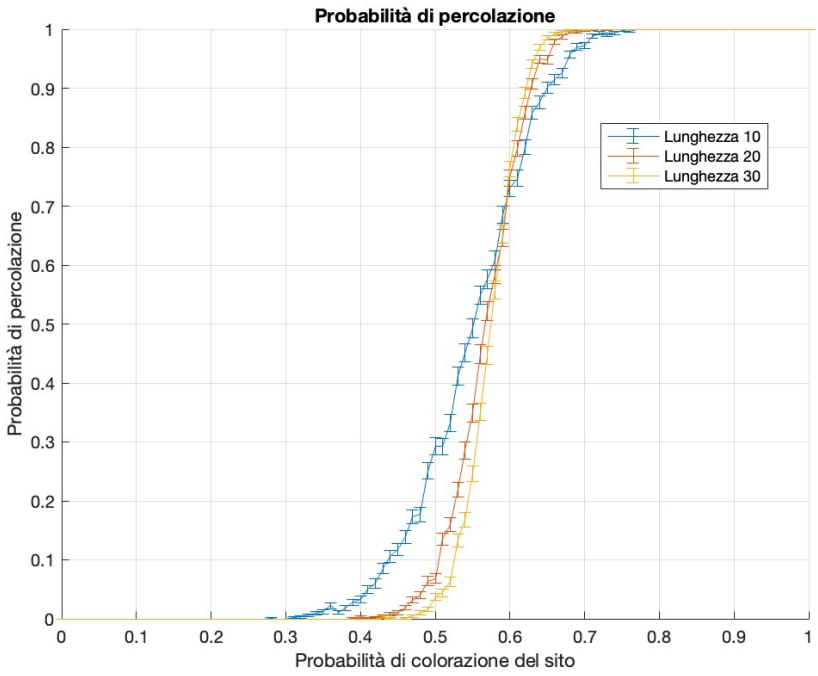


Figure 9: Grafico della soglia di percolazione al variare della taglia con errore

A questo benchmark è stato aggiunto un controllo che verifica la coerenza tra le probabilità di percolazione nelle direzioni top-bottom (TB) e left-right (LR), al fine di analizzare se le probabilità siano equivalenti. Per realizzare questo controllo, è stato utilizzato il seguente codice MATLAB all'interno dello script.

```
probSuccessTB = mean(successTB);
probSuccessLR = mean(successLR);
max = max(probSuccessTB, probSuccessLR);
threshold = 0.1; % Tolleranza

if ((probSuccessTB - probSuccesLR)/max) > threshold
    warning("Risultati diversi per taglia " + LL(idx) + " e
       prob " + p(i));
end
```

Dove:

- **max** rappresenta il valore massimo tra le probabilità di percolazione nelle direzioni TB e LR, utilizzato per calcolare la differenza relativa tra le due probabilità e verificarne la coerenza rispetto alla tolleranza fissata.
- **probSuccessTB** e **probSuccessLR** sono le probabilità medie di percolazione nelle direzioni top-bottom (TB) e left-right (LR), rispettivamente.

Bisogna considerare che il risultato di questo controllo sarà fortemente influenzato dal numero di misurazioni (**m**) e dalla configurazione del problema. Dai test eseguiti con lo script è possibile vedere che i warning si riducono all'aumentare dei siti colorati in quanto all'aumentare della probabilità di quest'ultimo avremo cluster più interconnessi.

3.1 Analisi di ulteriori osservabili

Ora andremo ad analizzare ulteriori osservabili con il relativo grafico che ci faranno comprendere meglio il valore della soglia di percolazione.

Prima osservabile. La **prima osservabile** è definita dalla seguente formula:

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

dove:

- s_{max} rappresenta la dimensione del cluster più grande, ossia il numero di siti che compongono il cluster di maggiore estensione.
- L è la lunghezza di un lato del reticolo quadrato, quindi L^2 è il numero totale di siti nel reticolo.

A livello probabilistico questa formula rappresenta la **probabilità che un sito scelto a caso appartenga al cluster più grande**. In altre parole, è la frazione di siti che fanno parte del cluster massimo.

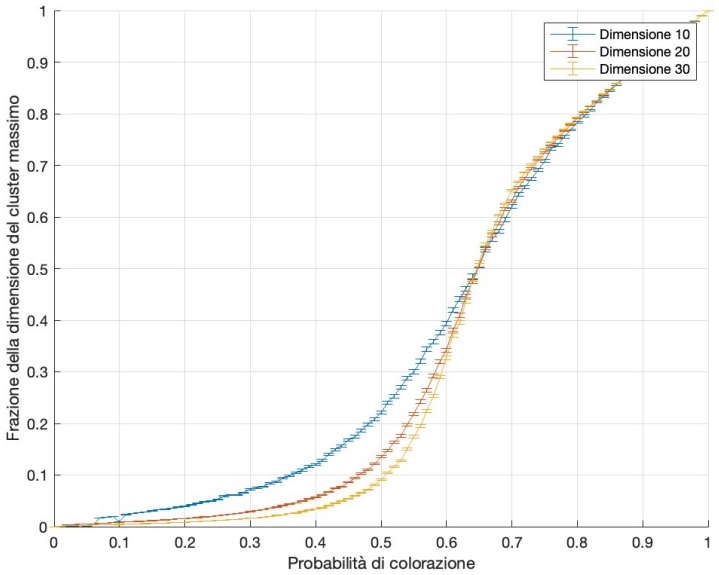


Figure 10: Evoluzione della probabilità che un sito casuale appartenga al cluster s_{max}

Seconda Osservabile. La **seconda osservabile** è definita dalla seguente formula:

$$P_1 = \frac{s_{max}}{pL^2}$$

dove:

- s_{max} come prima rappresenta il cluster più grande
- pL^2 è il numero medio di siti colorati che ci si aspetta di trovare sul reticolo.

Sostanzialmente permette di visualizzare, come è possibile vedere in Figura 11, la probabilità che un sito appartenga al cluster massimo sapendo che è colorato.

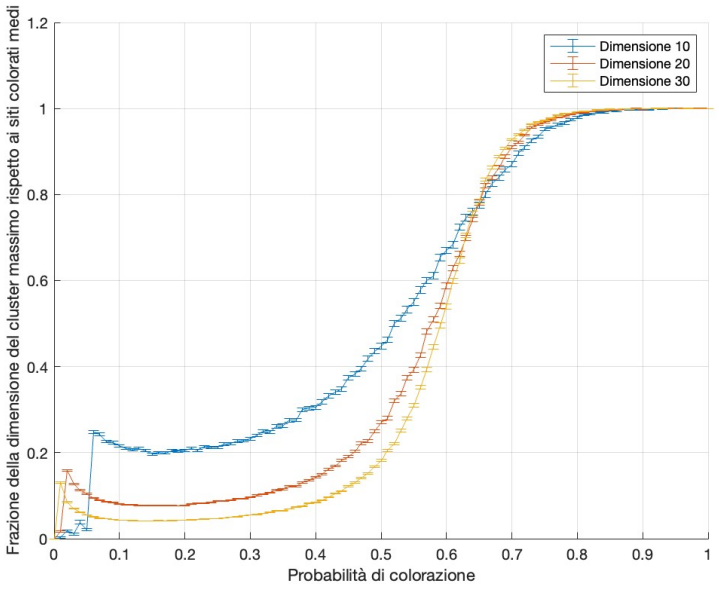


Figure 11: Evoluzione probabilità che un sito colorato appartenga a s_{max}

Terza osservabile. La **terza osservabile** è definita dalla seguente formula:

$$P_3 = \frac{s_{max}}{\sum_s s n_s}$$

dove:

- $\sum_s s n_s$ indica la sommatoria di ogni taglia possibile di un cluster (s) con le sue occorrenze per il reticolo preso in esame (n_s)

Come è possibile vedere in Figura 12, questo rapporto esprime la frazione occupata dal cluster massimo considerando i siti colorati effettivi.

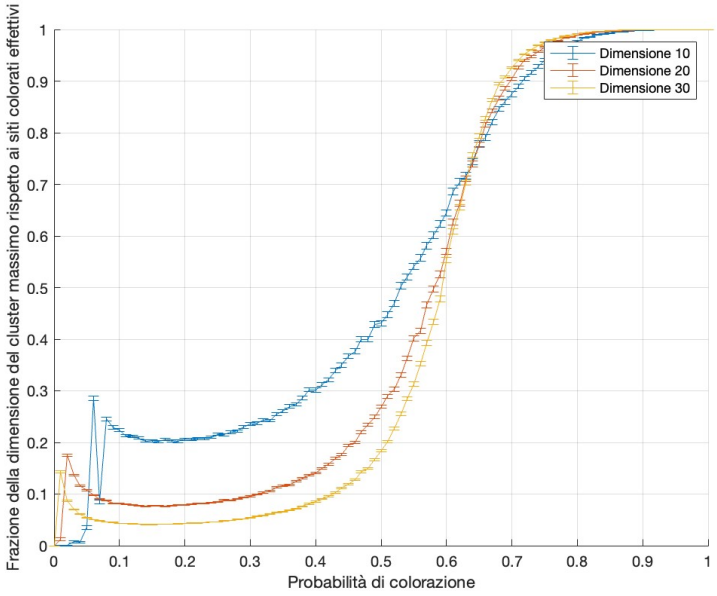


Figure 12: Cluster massimo in rapporto ai siti colorati effettivi con errore

Quarta osservabile. La **quarta osservabile** è descritta dalla seguente formula:

$$RACS = \frac{\sum_{s \neq s_{max}} s(s n_s)}{\sum_{s'} s' n_{s'}}$$

dove:

- $RACS$ è l'acronimo di Reduced Average Cluster Size, ovvero "taglia media dei cluster ridotta".
- Nel **numeratore** viene escluso il valore s_{max}; di conseguenza, la media calcolata risulta ridotta.

Questa osservabile permette di considerare la dimensione media, in rapporto con la taglia, dei cluster diversi da quello più grande (s_{max}). Infatti, come si può vedere dall'andamento in Figura 13, i valori si concentrano attorno alla soglia di percolazione.

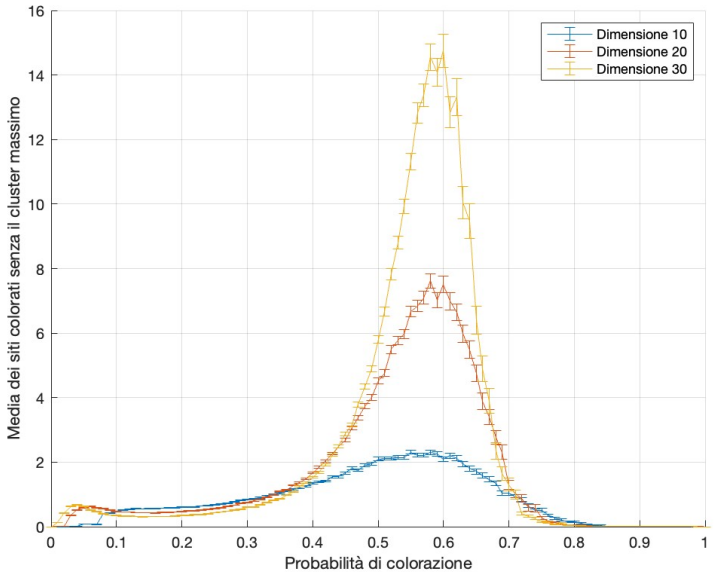


Figure 13: Media della dimensione dei siti senza considerare il cluster massimo con errore

3.2 Osservazioni sulla soglia

Le osservabili analizzate ci hanno consentito di studiare, da prospettive diverse, il comportamento del sistema intorno al valore della soglia di percolazione, evidenziando fenomeni interessanti. Le prime tre osservabili mostrano un andamento che tende a una curva sigmoide, simile a quanto osservato durante l'analisi della soglia di percolazione. La RACS fornisce informazioni significative sul comportamento del sistema. Analizzando il grafico (Figura 13), si osserva che, al di sotto della soglia di percolazione, i cluster sono di piccole dimensioni. Al di sopra della soglia, il cluster percolante diventa dominante, mentre i cluster rimanenti tornano a essere piccoli. In prossimità della soglia, invece, si registra una maggiore varietà nelle dimensioni dei cluster, portando a una media (escludendo il cluster più grande) più elevata. Questo massimo della RACS rappresenta un chiaro indicatore della transizione di fase.

Si può dunque osservare che il valore massimo della RACS coincide con la soglia di percolazione, la quale, tenendo conto dell'errore, assume il seguente valore:

$$0.5833 \pm 0.0167$$

che per valori grandi della taglia, considerando anche le limitazioni della macchina sulla quale sono state eseguite le simulazioni, si afferma ad un valore che si aggira a 0.6.

Da ciò possiamo fare considerazioni in merito a cosa significa **percolazione in un reticolo di taglia infinita**: implica che esiste un cluster che si estende per tutto il reticolo. Questo accade quando la probabilità di occupazione dei siti, p , supera un certo valore critico, p_c. Superata tale soglia, la probabilità che un sito appartenga al cluster più grande diventa uguale a 1, segnalando la presenza di un cluster infinito.