Percolazione nei reticoli quadrati

Studente: Alessio Russo **Matricola:** 376856
Corso di studio: Scienze Informatiche **Esame:** Modellazione e Simulazioni Numeriche

1 Algoritmo di Hoshen-Kopelman

L'algoritmo di Hoshen-Kopelman (HK76) è una tecnica di etichettatura multipla dei cluster. Il reticolo viene visitato sito per sito per colonne, partendo dallo spigolo in alto a sinistra per arrivare a quello in basso a destra. Si prenda, ad esempio, il reticolo in Figura 1

0	1	0	0	1	1	0	0	0	1	0	1	0	1	0
0	1	1	0	0	0	1	0	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	0	0	1	0	0	0
1	0	1	0	1	0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	0	1	0	0	0	1	0	0	1	0
0	0	1	1	1	1	1	0	1	1	1	1	1	0	1
0	0	1	1	1	0	0	0	0	1	1	1	1	0	1
0	1	0	1	1	1	1	0	1	1	1	1	0	1	0
0	1	0	0	1	1	0	0	1	1	1	1	0	0	1
1	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	1	0	1	0	1	1	0	1	1	1	1
1	1	0	0	0	1	0	1	0	1	1	1	1	1	1
1	1	1	0	0	1	1	0	1	0	0	1	0	0	0
0	1	0	1	1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	0	0	0	0	1	1	1	1	1	1	0

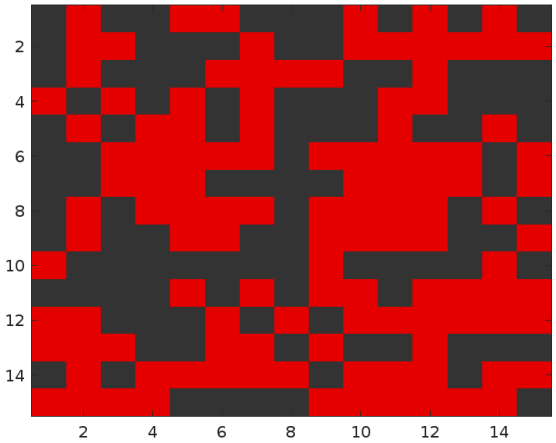

Figura 1: Esempio di reticolo quadrato 15×15 con siti colorati e non colorati

Durante la visita del reticolo, quando si incontra un sito colorato, allora: **(1)** Se il sito non è connesso ad altri siti colorato sopra o a sinistra, si inizia un nuovo cluster, a cui viene assegnata una `label` **(2)** Se c'è un primo vicino sopra o a sinistra colorato (uno solo dei due), il sito viene aggiunto al cluster del primo vicino colorato **(3)** Se i suoi primi vicini sono entrambi colorati, ma appartengono allo stesso cluster, il sito viene aggiunto al cluster dei primi vicini **(4)** Se i suoi primi vicini sono entrambi colorati, e non appartengono allo stesso cluster, il sito viene aggiunto al cluster con la `label` minore.

0	1	0	0	2	2	0	0	0	3	0	4	0	5	0
0	1	1	0	0	0	6	0	0	3	3	3	3	3	3
0	1	0	0	0	7	6	6	6	0	0	3	0	0	0
8	0	9	0	10	0	6	0	0	0	11	3	0	0	0
0	12	0	13	10	0	6	0	0	0	11	0	0	14	0
0	0	15	13	10	10	6	0	16	16	11	11	11	0	17
0	0	15	13	10	0	0	0	0	16	11	11	11	0	17
0	18	0	13	10	10	10	0	19	16	11	11	0	20	0
0	18	0	0	10	10	0	0	19	16	11	11	0	0	21
22	0	0	0	0	0	0	0	19	0	0	0	0	23	0
0	0	0	0	24	0	25	0	19	19	0	26	26	23	23
27	27	0	0	0	28	0	29	0	19	19	19	19	19	19
27	27	27	0	0	28	28	0	30	0	0	19	0	0	0
0	27	0	31	31	28	28	28	0	32	32	19	0	33	33
34	27	27	27	0	0	0	0	35	32	32	19	19	19	0

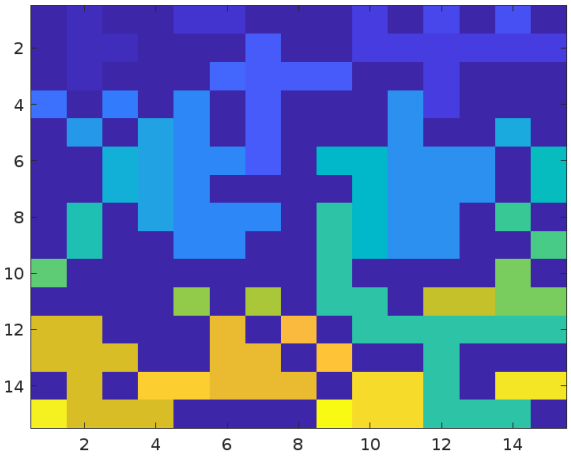

Figura 2: Esempio di etichettatura dei cluster con l'algoritmo di Hoshen-Kopelman

Ad esempio, il cluster associati al reticolo in Figura 1 sono mostrati in Figura 2.

Tuttavia, quando si incontra un caso come quello descritto nel punto **(4)**, occorre memorizzare che i due cluster sono in realtà lo stesso cluster. Questo viene fatto usando un vettore chiamato **Label of Label** (`LofL`), che contiene tutta l'informazione necessaria sui label dei cluster. In particolare, il modulo `HKclass`: per un *good label*, memorizza la taglia del cluster; per *bad label*, memorizza qual è il vero cluster label a cui questo label appartiene. Questa distinzione viene fatta attraverso i segni dei numeri interi contenuti in LofL. Di seguito è riportato il LofL corrispondete al reticolo preso in esame

ID	1	2	3	4	5	6	7	8	9	10	11	12
Val	4	2	56	-3	-3	24	-6	1	1	-6	-3	1

ID	13	14	15	16	17	18	19	20	21	22	23	24
Val	-10	1	-10	-3	2	2	-3	1	1	1	-3	1

ID	25	26	27	28	29	30	31	32	33	34	35
Val	1	-3	18	-27	1	1	-27	-3	-3	-27	-3

Tuttavia, l'algoritmo HK restituisce in modo corretto le taglie dei cluster, ma non garantisce che tutti i siti di un fissato cluster abbiano lo stesso valore. Per questo motivo, effettuiamo una rietichettatura successiva. La Figura 3 ne mostra un esempio.

0	1	0	0	2	2	0	0	0	3	0	3	0	3	0
0	1	1	0	0	0	6	0	0	3	3	3	3	3	3
0	1	0	0	0	6	6	6	6	0	0	3	0	0	0
8	0	9	0	6	0	6	0	0	0	3	3	0	0	0
0	12	0	6	6	0	6	0	0	0	3	0	0	14	0
0	0	6	6	6	6	6	0	3	3	3	3	3	0	17
0	0	6	6	6	0	0	0	0	3	3	3	3	0	17
0	18	0	6	6	6	6	0	3	3	3	3	0	20	0
0	18	0	0	6	6	0	0	3	3	3	3	0	0	21
22	0	0	0	0	0	0	0	3	0	0	0	0	3	0
0	0	0	0	24	0	25	0	3	3	0	3	3	3	3
27	27	0	0	0	27	0	29	0	3	3	3	3	3	3
27	27	27	0	0	27	27	0	30	0	0	3	0	0	0
0	27	0	27	27	27	27	27	0	3	3	3	0	3	3
27	27	27	27	0	0	0	0	3	3	3	3	3	3	0

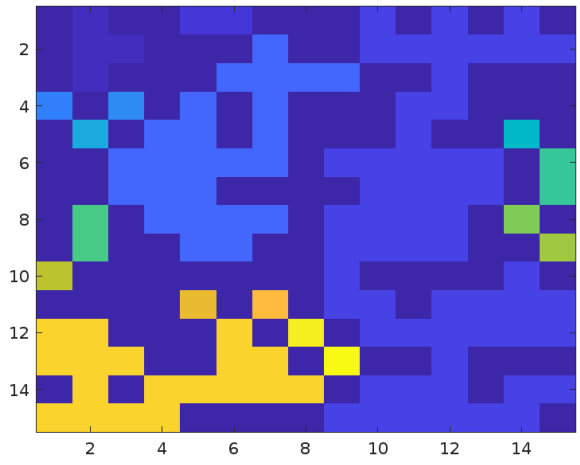

Figura 3: Esempio di rietichettatura dei cluster con etichette uniformi per ciascun gruppo

A questo punto, l'obiettivo è determinare se esistono cluster percolanti all'interno del reticolo, ossia se esiste almeno un'etichetta condivisa tra la prima e l'ultima riga (percolazione verticale) e tra la prima e l'ultima colonna (percolazione orizzontale). Per fare ciò, possiamo sviluppare un algoritmo che, basandosi sull'estrazione delle etichette **uniche** presenti lungo i bordi della matrice, e mediante l'utilizzo dell'operazione `intersect`, verifica l'esistenza di almeno una etichetta comune tra i bordi opposti. Se tale etichetta è presente, viene restituito `true` per il tipo di percolazione considerato, altrimenti `false`.

Si noti che, poiché ci interessa esclusivamente determinare il cluster di appartenenza della prima e dell'ultima riga e colonna per valutare la percolazione, è sufficiente rietichettare solo questi elementi, ignorando il centro del reticolo e risparmiando così tempo di calcolo. La Figura 4 ne mostra un esempio.

Concludiamo dicendo che l'analisi della correttezza dell'implementazione proposta è stata effettuata non soltanto tramite test individuali sul singolo algoritmo, ma anche attraverso un confronto diretto

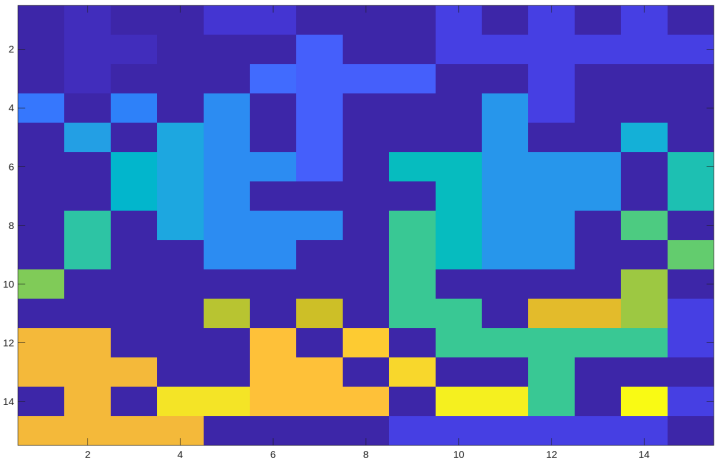

Figura 4: Esempio di rietichettatura dei soli bordi per il test di percolazione

con l'algoritmo naive presentato a lezione. Nello specifico, è stato generato un reticolo quadrato di taglia 100, con una probabilità di colorazione dei siti pari al 60%. Tale reticolo è stato analizzato prima con l'algoritmo naive e successivamente con l'algoritmo HK76, confrontando i risultati ottenuti per la percolazione verticale (top-bottom) e orizzontale (left-right). Questo confronto è stato ripetuto 10.000 volte, e in tutti i casi i risultati forniti dai due algoritmi sono stati coincidenti.

2 Confronto tra algoritmi naive e HK76

L'algoritmo di Hoshen-Kopelman (HK76, indicato in *blu*) ha mostrato una significativa efficienza computazionale superiore rispetto all'algoritmo di etichettatura naive (indicato in *arancione*), precedentemente implementato. Questa valutazione è stata condotta attraverso due diverse analisi: la prima ha considerato il tempo di calcolo mantenendo costante la probabilità di colorazione dei siti e variando la dimensione del reticolo, mentre la seconda ha studiato il comportamento opposto, ovvero il tempo di calcolo mantenendo costante la dimensione del reticolo e variando la probabilità di colorazione dei siti.

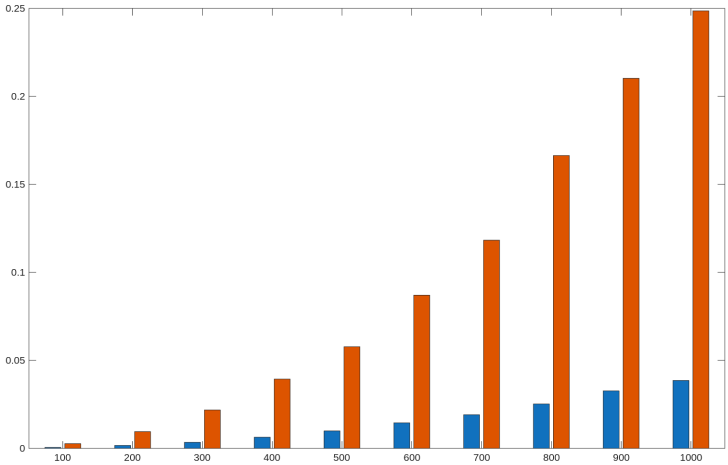

Figura 5: Tempo di esecuzione in funzione della dimensione del reticolo per gli algoritmi HK76 e naive.

Nel primo caso, è stata fissata una probabilità di colorazione dei siti pari al 60%, variando la dimensione del reticolo quadrato nell'intervallo compreso tra 100 e 1000, con incrementi di 100, per un totale di 10 configurazioni. La Figura 5 mostra i tempi medi di esecuzione dell'algoritmo, calcolati eseguendo ciascuna configurazione 50 volte. Si noti come, all'aumentare della dimensione del reticolo, l'algoritmo HK76 offra prestazioni migliori rispetto all'algoritmo naive in termini di tempo di esecuzione. Di seguito sono riportati i tempi medi di esecuzione:

hk76	0.0005	0.0018	0.0037	0.0065	0.0099	0.0140	0.0192	0.0255	0.0313	0.0385
naive	0.0023	0.0090	0.0199	0.0354	0.0552	0.0806	0.1097	0.1482	0.1877	0.2383

e i relativi errori (da moltiplicare per 10^{-3}):

hk76	0.0107	0.0511	0.0261	0.0830	0.1095	0.1083	0.1271	0.1536	0.1598	0.1649
naive	0.0108	0.0224	0.0177	0.1090	0.1216	0.2112	0.2641	0.2072	0.3270	0.4488

Nel secondo caso, invece, la dimensione del reticolo è stata mantenuta fissa a 500, mentre la probabilità di colorazione dei siti è stata variata da 0,1 a 1, con incrementi di 0,1, per un totale di 10 esperimenti.

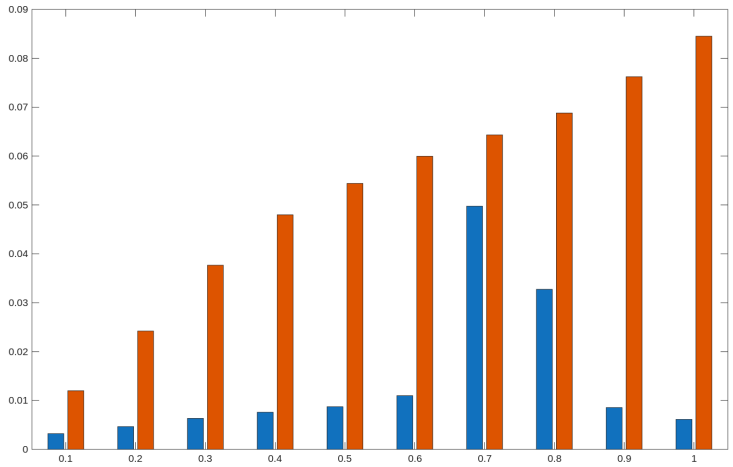

Figura 6: Tempo di esecuzione in funzione della probabilità di colorazione per gli algoritmi HK76 e naive.

La Figura 6, che mostra i tempi medi di esecuzione su 50 iterazioni, evidenzia come l'algoritmo HK76 risulti nettamente migliore rispetto alla versione naive. Di seguito sono riportati i tempi medi di esecuzione:

hk76	0.0032	0.0047	0.0063	0.0076	0.0087	0.0110	0.0497	0.0327	0.0086	0.0061
naive	0.0120	0.0242	0.0377	0.0480	0.0544	0.0600	0.0643	0.0688	0.0762	0.0845

e i relativi errori (da moltiplicare per 10^{-3}):

hk76	0.0776	0.0643	0.0935	0.0997	0.0864	0.1096	0.6884	0.4468	0.0981	0.0650
naive	0.1488	0.2100	0.4623	0.4539	0.3350	0.4271	0.2585	0.2870	0.2396	0.3537

3 Ricerca della soglia di percolazione