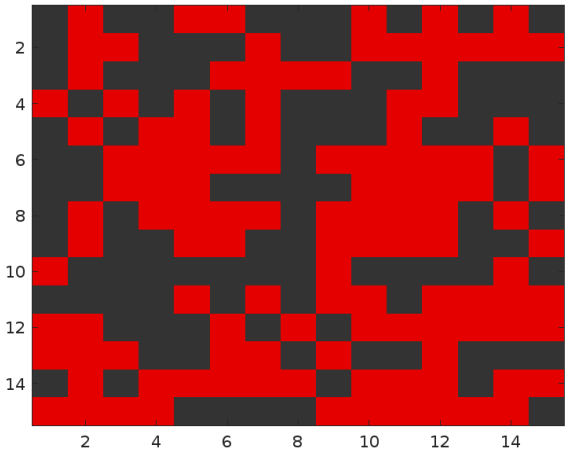
Percolazione nei reticoli quadrati bidimensionali

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

1 Algoritmo di Hoshen-Kopelman

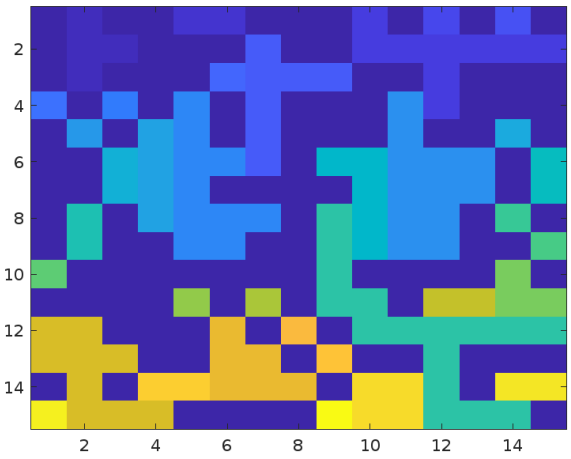
L'algoritmo di Hoshen-Kopelman (HK76) è una tecnica di etichettatura multipla dei cluster. Il reticolo viene visitato sito per sito per colonne, partendo dallo spigolo in alto a sinistra per arrivare a quello in basso a destra. Si prenda, ad esempio, il reticolo in figura

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



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. Ad esempio, il cluster associati al reticolo precedente sono mostrati in figura

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



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

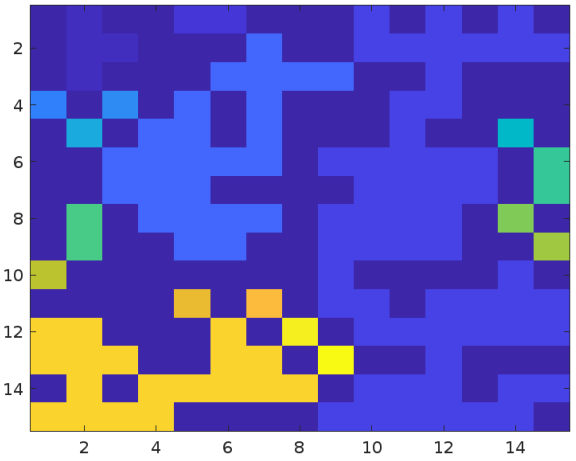
ID	1	2	3	4	5	6	7	8	9	10	11	12
Val	4	2	56	-3	-3	24	-6	1	1	-6	-3	1

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, al fine di identificare la presenza di cluster percolanti, effettuiamo una rilabelizzazione successiva. Di seguito ne è mostrato 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



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`.

Tuttavia, poiché ci interessa esclusivamente determinare il cluster di appartenenza della prima e dell'ultima riga e colonna per valutare la percolazione, è sufficiente rilabelare solo questi elementi, ignorando il centro del reticolo e risparmiando così tempo di calcolo. La Figura 1 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 con l'algoritmo naive presentato a lezione. Nello specifico, è stato generato un reticolo bidimensionale di dimensioni 100×100, con una probabilità di colorazione dei siti pari al 50%. Tale reticolo è stato analizzato prima con l'algoritmo naive e successivamente con l'algoritmo HK76, confrontando i risul-

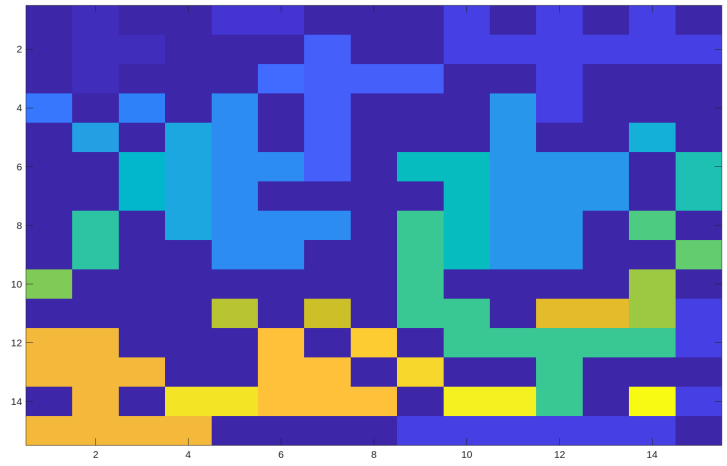


Figura 1: Esempio di ri-labeling limitato ai bordi del reticolo

tati 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.

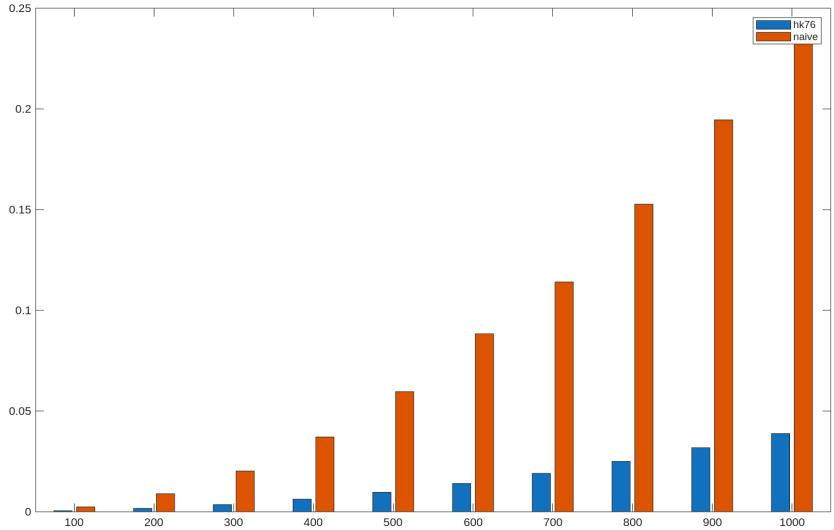


Figura 2: Tempo di esecuzione in funzione della dimensione del reticolo per gli algoritmi HK76 e naive.

Nel primo caso, è stata fissata una probabilità di colorazione dei siti pari al 50%, variando la dimensione del reticolo bidimensionale nell'intervallo compreso tra 1000 e 10000, con un incremento di 1000, per un totale di 10 configurazioni. La figura 2 mostra che, all'aumentare della dimensione del reticolo, l'algoritmo HK76 offre prestazioni migliori rispetto all'algoritmo naive in termini di tempo di esecuzione.

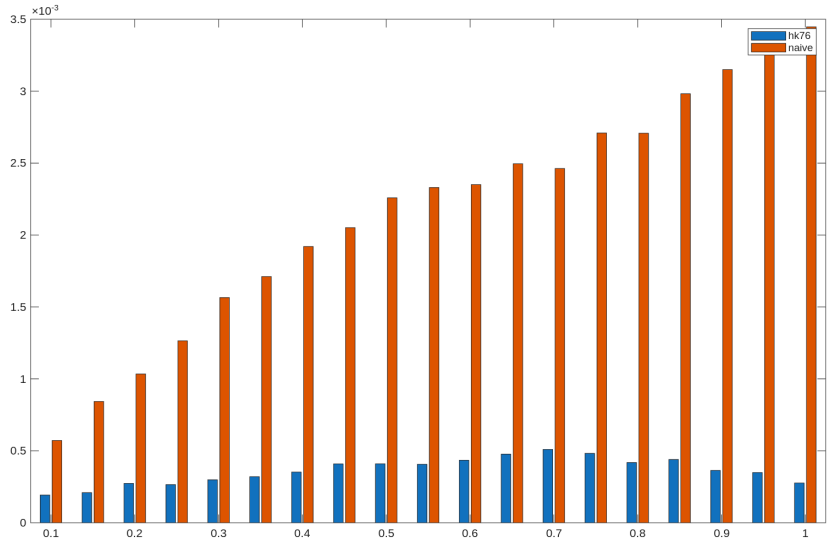


Figura 3: Tempo di esecuzione in funzione della probabilità di colorazione per gli algoritmi HK76 e naive.

Nel secondo caso, invece, la dimensione del reticolo è stata mantenuta fissa a 1000×1000, 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. La figura 3 evidenzia come l'algoritmo naive risulti più efficiente rispetto all'algoritmo HK76 quando la probabilità di colorazione è bassa. Tuttavia, con l'aumentare della probabilità di colorazione, l'algoritmo HK76 migliora nettamente le proprie prestazioni, risultando significativamente più veloce.

3 Ricerca della soglia di percolazione