Introduzione alla teoria della percolazione

Leonardo Ongari *Università degli studi di Parma*
Cremona, Italia
leonardo.ongari@studenti.unipr.it

***Sommario*—La teoria della percolazione si occupa di sviluppare modelli matematici per descrivere il fenomeno fisico della percolazione, caratterizzato dallo scorrimento di un *fluido* all'interno di un *mezzo*. In questa relazione vengono mostrate le caratteristiche principali dei modelli, le nozioni teoriche associate ed alcune analisi sugli algoritmi utilizzabili per effettuare simulazioni consistenti.**

***Keywords*—Percolazione, Modelli, Simulazioni**

I. Introduzione

La teoria della percolazione nasce con l'obiettivo di ottenere un modello matematico per descrivere il fenomeno fisico della percolazione. Questo fenomeno descrive lo scorrimento di un fluido all'interno di un materiale tipicamente poroso. È importante specificare che il significato del termine "percolazione" può riferirsi a diversi contesti, a seconda del campo in cui si sta operando. Alcuni esempi sono:

- un soluto che diffonde attraverso un solvente;
- elettroni che migrano attraverso un reticolo atomico;
- molecole che penetrano un solido poroso;
- una malattia che infetta una comunità.

La formalizzazione matematica del problema ha reso possibile la creazione di un *modello*. Un punto chiave molto importante di un modello è il concetto di *astrazione*, caratteristica che permette di operare senza considerare alcune variabili dipendenti dal contesto [1].

II. Background

Un metodo efficace per l'astrazione del concetto consiste in una rappresentazione tramite un reticolo, in inglese "lattice".

Definizione 1 (Reticolo)**.** Un reticolo L di dimensione n è una coppia $\langle S, B\rangle$, dove:

- $S = \{s_i : i \in \mathbb{N},\, 0 \leq i < n\}$ è l'insieme dei siti;
- $B = \{(s_i, s_j) \in S^2 : s_i \text{ e } s_j \text{ siano primi vicini}\}$ è l'insieme dei legami.

In un reticolo in cui vale questa relazione è possibile descrivere due tipi di percolazione:

- percolazione di legame;
- percolazione di sito.

Percolazione di legame

È la prima versione del modello fornita da Broadbent e Hammersley [1]. Ogni legame ha probabilità p di essere "aperto", quindi probabilità $1-p$ di essere "chiuso". Se due siti formano un legame aperto, vi è una connessione diretta tra i due. Al contrario, un legame chiuso elimina la connessione. In questa versione, "avviene percolazione" se esiste un percorso (insieme di connessioni dirette) che attraversa l'intero reticolo. La percolazione può verificarsi sia in verticale (alto-basso) sia in orizzontale (sinistra-destra).

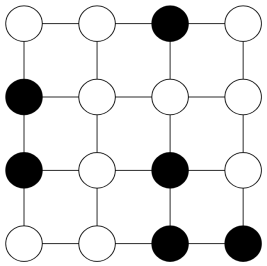

Figura 1: Esempio di reticolo quadrato bidimensionale

Percolazione di sito

In questo modello ogni sito ha una probabilità p di essere occupato, di conseguenza probabilità $1-p$ di essere vuoto. In figura 1 viene mostrato un reticolo bidimensionale quadrato con nodi occupati (neri) e vuoti (bianchi). Questo è il modello che verrà utilizzato per lo studio dell'argomento e dei vari algoritmi e verrà approfondito in dettaglio nella sezione III.

Soglia di percolazione

I due modelli appena introdotti rappresentano soluzioni valide per lo studio del fenomeno. Nonostante la somiglianza, vi sono differenze concettuali che si riflettono anche nel calcolo di alcuni valori caratteristici [2], [3], [4].

Definizione 2 (Soglia di percolazione)**.** Sia L un reticolo di dimensione infinita[1]. Sia p la probabilità di occupazione di un sito o di apertura di un legame, a seconda del modello scelto. Sia p uguale per ogni elemento del reticolo. La soglia di percolazione per L è definita come la probabilità p_c tale per cui:

- se $p > p_c$, allora vi è percolazione;
- se $p < p_c$, allora non vi è percolazione.

È possibile visualizzare il concetto di soglia nel grafico mostrato in figura 2, in cui l'asse delle ascisse è associato alla probabilità p, mentre l'asse delle ordinate è associato alla probabilità che avvenga percolazione p_{perc}.

[1]Con il termine "infinito" si fa riferimento all'estensione intuitiva delle varie proprietà della struttura, come avviene in matematica per il concetto di *limite all'infinito*.

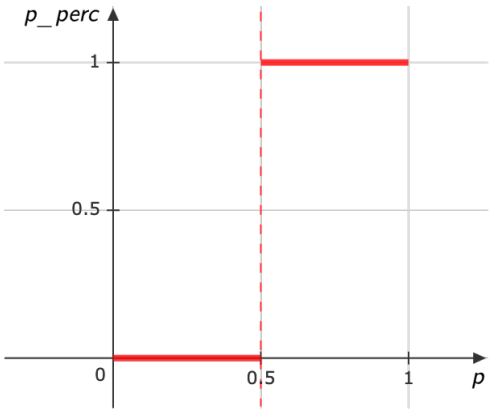

Figura 2: Grafico relativo alla soglia di percolazione ($p_c = 0.5$)

Nei casi reali, ovvero per reticoli di taglia finita, non è possibile stabilire un'affermazione così forte. Ciò che è possibile preservare dalla definizione è che esiste un **punto critico** p_c relativo alla probabilità p, oltre al quale è più probabile che avvenga percolazione e, al contrario, al di sotto del quale è meno probabile che questo si verifichi.

Lattice	p_c **(Site)**	p_c **(Bond)**
Cubic (body-centered)	0.246	0.1803
Cubic (face-centered)	0.198	0.119
Cubic (simple)	0.3116	0.2488
Diamond	0.43	0.388
Honeycomb	0.6962	0.65271*
4-Hypercubic	0.197	0.1601
5-Hypercubic	0.141	0.1182
6-Hypercubic	0.107	0.0942
7-Hypercubic	0.089	0.0787
Square	0.592746	0.50000*
Triangular	0.50000*	0.34729*

Tabella I: Punti critici (p_c) per reticoli regolari.

In letteratura sono presenti diversi studi sulle caratteristiche di vari reticoli e i rispettivi valori di soglia. La tabella I mostra i punti critici per diversi reticoli regolari, ovvero composti da elementi ripetuti della stessa forma. La colonna **Lattice** indica la forma del reticolo, mentre le colonne **Site** e **Bond** distinguono i valori in percolazione di sito e di legame, rispettivamente [5]. Vi è una lieve, ma evidente, discrepanza tra i valori nelle due colonne. In generale, la rappresentazione tramite occupazione dei siti è considerata più generica rispetto alla sua controparte, questo perché la percolazione di legame può essere riformulata in termini di percolazione di sito, ma non si può affermare il contrario. I valori affiancati da un asterisco possono essere trovati tramite calcoli analitici, grazie ad alcune caratteristiche della forma del reticolo, sono quindi considerati *conosciuti*. È interessante notare che la tabella è composta per lo più da valori non conosciuti, cioè valori ottenuti da simulazioni al calcolatore.

Cluster-finding

Nel modello che opera sui siti, la percolazione viene rilevata in seguito a un processo di *cluster-finding*, che consiste nel partizionare il reticolo in diverse classi, tramite una relazione di equivalenza.

Definizione 3. Una relazione di equivalenza R su un insieme A è una relazione binaria che gode delle seguenti proprietà:

- $\forall x \in A : xRx$ (riflessività);
- $\forall x, y \in A : xRy \rightarrow yRx$ (simmetria);
- $\forall x, y, z \in A : xRy \wedge yRz \rightarrow xRz$ (transitività).

La relazione utilizzata è strettamente collegata al concetto di primi vicini. La definizione di primi vicini per un sito può variare a seconda della tipologia (forma e dimensioni) del reticolo utilizzato.

Distribuzione binomiale

Per formalizzare in modo completo il modello, è necessario introdurre la nozione di *variabile aleatoria*. Non verrà trattato l'argomento nel dettaglio, per lasciare più spazio alle implementazioni. In questo contesto, si introduce il concetto di variabile aleatoria come una variabile il cui valore dipende da un evento non deterministico. Questo evento è descritto attraverso funzioni di densità specifiche, che seguono leggi di *distribuzione* della probabilità [6]. Esistono diverse tipologie di distribuzioni, tra cui quella binomiale che, riassumendo, descrive un esperimento di prove ripetute.

Questo tipo di distribuzione è caratterizzato da una funzione di densità a 2 parametri

$$\mathcal{B}_{n,p}(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{1}$$

dove n rappresenta il numero di prove e p la probabilità di successo di ciascuna. La variabile k indica il risultato di cui vogliamo verificare la probabilità. L'equazione II rappresenta *" la probabilità che si ottengano k successi ripetendo n volte una prova con probabilità di successo p "*.

III. Implementazione

Prima soluzione

In questa relazione verranno discusse proprietà e risultati dell'algoritmo Hoshen-Kopelman [7]. Tuttavia, per dimostrarne la correttezza e provarne l'efficienza, verranno effettuati dei confronti con una versione personalizzata, che chiameremo algoritmo A, la cui correttezza è ben nota data la sua semplicità. Il Cod. 1 mostra una prima soluzione al problema del partizionamento. È composto da 2 iterazioni principali: una "esterna" per tutti i siti colorati, e una "interna" per i siti adiacenti (primi vicini). Quest'ultima si appoggia ad un vettore che funge da *pila*, le cui dimensioni variano in maniera dinamica. Lo scopo della pila è quello di aggiungere, ad ogni iterazione interna, i siti occupati adiacenti al nodo corrente, per essere poi processati singolarmente. Ragionando sul funzionamento del codice, ci si convince piuttosto facilmente della ridondanza causata dalla pila. In altri termini, un sito occupato può essere processato più volte:

- una e una sola volta nell'iterazione esterna;
- una volta per ogni vicino occupato;
- altre volte per eventuali "catene di vicinanze".

Algoritmo 1 Pseudocodice dell'algoritmo A

```
largest_label ← 1
label ← zeros[n_columns, n_rows]
for [x, y] ← [0, 0] to n_columns, n_rows do
    if [x, y] = 1 ∧ label[x, y] = 0 then
        pila ← [x, y]
        label[x, y] ← largest_label
        j ← 1
        while j ≤ length(pila) do
            elem ← pila[j]
            x ← elem[1], y ← elem[2]
            left ← [x − 1, y]
            right ← [x + 1, y]
            down ← [x, y + 1]
            top ← [x, y − 1]
            if left = 1 ∧ label[left] = 0 then
                append left to pila
                label[left] ← largest_label
            end if
            if right = 1 ∧ label[right] = 0 then
                append right to pila
                label[right] ← largest_label
            end if
            if up = 1 ∧ label[up] = 0 then
                append up to pila
                label[up] ← largest_label
            end if
            if down = 1 ∧ label[down] = 0 then
                append down to pila
                label[down] ← largest_label
            end if
            j ← j + 1
        end while
        largest_label ← largest_label + 1
    end if
end for
```

Algoritmo 2 Pseudocodice dell'algoritmo Hoshen-Kopelman

```
largest_label ← 0
label ← zeros[n_columns, n_rows]
Lofl ← [0 : n_columns]

for [x, y] ← [0, 0] to [n_columns, n_rows do
    if [x, y] = 1 then
        left ← label[x − 1, y]
        above ← label[x, y − 1]
        if (left = 0) ∧ (above = 0) then
            largest_label ← largest_label + 1
            tmp ← largest_label
        else if (left = 1) ∧ (above = 0) then
            tmp ← left
        else if (left = 0) ∧ (above = 1) then
            tmp ← above
        else
            tmp ← min{left, above}
            bad_label ← max{left, above}
            UNION(bad_label, tmp)
        end if
        label[x, y] ← tmp
    end if
end for

procedure UNION(x, y)
    Lofl[FIND(x)] ← FIND(y)
end procedure

function FIND(x)
    while Lofl[x] ≠ x do
        x ← Lofl[x]
    end while
    return x
end function
```

Paradigma Union-Find

Come appena accennato, in questa relazione viene presentato l'algoritmo Hoshen-Kopelman, che fa parte della famiglia di algoritmi *union-find*, utilizzati nel calcolo di componenti connesse e, in questo caso, per il processo di cluster-finding. Questo paradigma prevede l'utilizzo di due procedure principali, che ne compongono il nome:

- *Union*: effettua una fusione tra due insiemi disgiunti quando ci si accorge che appartengono in realtà alla stessa classe di equivalenza;
- *Find*: determina a quale insieme appartiene l'elemento in elaborazione.

Nel Cod. 2 vengono mostrati i passaggi per l'implementazione del paradigma nel contesto della percolazione. Rispetto all'algoritmo iniziale, rimane un'iterazione sui siti per verificarne l'occupazione, ma appare diversa la logica applicata. In particolare, scompare l'utilizzo della pila con la conseguente ridondanza e, per ogni sito occupato, vengono considerati solo due primi vicini: sopra ($above$) e a sinistra ($left$). Se nessuno dei due vicini è occupato, al sito viene assegnata una nuova etichetta ($label$), aggiornata tramite un contatore; se soltato uno dei due vicini è occupato, il sito eredita la sua etichetta; se entrambi i vicini sono occupati e hanno la stessa etichetta, il comportamento è analogo al caso di un vicino occupato; se invece i vicini hanno etichette diverse, il sito eredita quella minore, prestando attenzione al fatto che in realtà le due facciano riferimento allo stesso cluster [8]. Infatti, in questo procedimento si possono avere siti di uno stesso cluster con etichette diverse (provare per credere!). Quest'ultimo punto è cruciale: è necessario tenere conto di questa informazione senza impattare negativamente sul costo computazionale dell'algoritmo; una rinomina totale delle etichette sarebbe infatti troppo costosa e si perderebbe parte del vantaggio ottenuto rispetto all'algoritmo A. Per coprire questo aspetto, per ogni cluster si sceglie una etichetta "di rappresentanza", che chiameremo *good label*. Ogni altra etichetta viene considerata una *bad label* e deve essere in qualche modo collegata alla rispettiva *good label*. Il metodo di collegamento scelto consiste nell'utilizzo di un array $Lofl$ (Label of label), che viene aggiornato durante l'esecuzione. A questo proposito, entrano in gioco le nuove procedure:

- *Union*: ha il compito di collegare una *bad label* ad una potenziale[2] *good label* e viene descritta facilmente in termini di *Find*;
- *Find*: viene implementata come iterazione di punto fisso della funzione rappresentata dall'array $Lofl$. In altri termini, si cerca in modo iterativo la condizione di arresto $Lofl[x] = x$.

[2]Con il termine "potenziale" si fa riferimento al fatto che, quando si effettua collegamento, non si può sapere se l'etichetta minore sia effettivamente una *good label*, perché potrebbe a sua volta essere collegata. Tuttavia, si può dire che tramite una catena di collegamenti è possibile risalire ad una *good label*.

RIFERIMENTI BIBLIOGRAFICI

[1] S. R. Broadbent and J. M. Hammersley, "Percolation processes: I. crystals and mazes," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 53, no. 3, pp. 629-641, 1957.

[2] C. Stover and E. W. Weisstein, "Bond percolation." [Online]. Available: https://mathworld.wolfram.com/BondPercolation.html

[3] ——, "Site percolation." [Online]. Available: https://mathworld.wolfram.com/SitePercolation.html

[4] ——, "Percolation threshold," *from Wolfram MathWorld*, 2002. [Online]. Available: https://mathworld.wolfram.com/

[5] D. Stauffer and A. Aharony, *Introduction to percolation theory*. Taylor & Francis, 2018.

[6] F. Edition, A. Papoulis, and S. U. Pillai, *Probability, random variables, and stochastic processes*. McGraw-Hill Europe: New York, NY, USA, 2002.

[7] J. Hoshen and R. Kopelman, "Percolation and cluster distribution. i. cluster multiple labeling technique and critical concentration algorithm," *Phys. Rev. B*, vol. 14, pp. 3438–3445, Oct 1976. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.14.3438

[8] T. Fricke, "The hoshen-kopelman algorithm," *URL: http://www. ocf. berkeley. edu/˜ fricke/projects/hoshenkopelman/hoshenkopelman. html*, 2004.