Simulazione Monte Carlo della ricerca del sito di legame da parte di un fattore di trascrizione

Davide Cois, Benedetta Mariani

Maggio 2019

Sommario

I fattori di trascrizione (TF) sono delle proteine che, per dare avvio alla trascrizione di un particolare gene, devono legarsi al loro specifico sito di legame sul DNA: il nostro programma si propone di simulare la *ricerca correlata* del sito di legame sul DNA da parte di un fattore di trascrizione, la quale consiste in una diffusione in una dimensione lungo la catena del DNA (*sliding*) e in una diffusione in tre dimensioni (*hopping*) in seguito alla dissociazione della proteina dalla catena del DNA. Abbiamo seguito il lavoro proposto nell'articolo [1] e i valori dei parametri indicati dagli autori per consentire un confronto con i dati sperimentali. In particolare, lo scopo del nostro programma è quello di calcolare il rate k con cui il TF raggiunge un sito di legame, chiamato *operatore*, in presenza di due possibili siti di legame, dividerlo per il rate k_0 ottenuto nel caso di un singolo *operatore*, e confrontare questo rapporto con quanto è ricavato dal fit dei dati sperimentali in [1], al variare della distanza tra gli operatori.

1 Modello di sliding e hopping

Definiamo con *sliding* la diffusione unidimensionale lungo la struttura ad elica del DNA, con *hopping* invece la diffusione in tre dimensioni. Descriviamo in seguito le regole che descrivono le probabilità di ogni evento:

Sliding. Un TF legato al DNA può compiere uno spostamento diffusivo lungo la catena del DNA verso la coppia di basi più vicina, in una qualsiasi direzione, oppure dissociarsi microscopicamente, e dare avvio quindi all'hopping. La prima situazione ha probabilità $P = s^2/(1+2s^2)$, e l'ultima $P_d = 1/(1+2s^2)$, in cui $s = \sqrt{\frac{D_1}{k_d^{micro}}}$ (D_1 è un coefficiente di diffusione unidimensionale e k_d^{micro} è un rate di dissociazione microscopica). s^2 risulta un parametro nel nostro modello.

Hopping. L'*hopping* ha luogo in seguito ad una *dissociazione microscopica* della proteina dal filamento del DNA. La proteina inizia allora la diffusione in 3D da una distanza r_{start} dal DNA fino a una distanza r, e in seguito torna ad una distanza r_{min} dall'asse del DNA, da cui si rilega al filamento tramite un'*associazione microscopica.* Matematicamente modellizziamo questo evento tramite la probabilità che un oggetto che parte da una distanza r_{start} da un cilindro di raggio r_{min} ritorni al cilindro, dopo aver fatto una

escursione ad una distanza maggiore o uguale ad r:

$$P(r) = \frac{\ln r_{start} - \ln r_{min}}{\ln r - \ln r_{min}} \tag{1}$$

Inoltre, con *dissociazione macroscopica* definiamo una dissociazione dal DNA ed escursione fino ad una distanza prefissata r_{max} o z_{max} (con z_{max} la distanza di persistenza lungo il DNA, presa pari a 23nm) senza alcuna riassociazione al DNA. Questo implica la perdita di correlazione con la catena da cui il TF si è dissociato. r_{max} è un parametro nel nostro modello.

L'escursione del TF ad una distanza r è combinata con uno spostamento lungo l'asse del DNA ed è caratterizzata dalle variabili (z, ϕ, t), che descrivono rispettivamente la distanza lungo l'asse del DNA, la variazione dell'angolo di legame tra eventi consecutivi di dissociazione e associazione, e il tempo di escursione. La distribuzione analitica di queste variabili in funzione di r non è nota. A tal fine, come prerequisito della simulazione principale (la ricerca del sito di legame da parte del TF), vengono effettuati dei random walk su un reticolo 2D quadrato, perpendicolare all'asse del DNA, con taglia della griglia pari a $r_{step} = r_{start}$ - r_{min}.
I random walk partono da una distanza r_{start} e terminano quando viene raggiunto r_{min}. Per ogni traiettoria che raggiunge (r, $r + r_{step}$), vengono registrati il numero di passi n complessivi della traiettoria, e l'angolo di legame ϕ, ovvero la variazione in angolo rispetto alla ultima dissociazione, e vengono dunque immagazzinati i dati in istogrammi per n e per ϕ.
Il numero di passi n effettuati nelle simulazioni in 2D equivale a $\frac{2}{3}$ dei passi che verrebbero effettuati in 3D. Quindi, la distanza z lungo l'asse del DNA coperta in una escursione viene calcolata secondo la distribuzione Gaussiana:

$$p(z, n) = \frac{1}{\sqrt{\pi n r_{step}^2}} \exp\left(-\frac{z^2}{n r_{step}^2}\right) \tag{2}$$

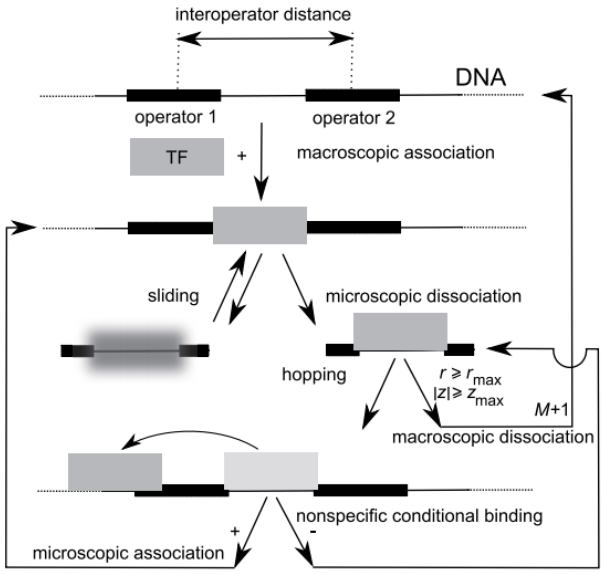
2 Modello della simulazione

L'algoritmo conta il numero M di dissociazioni macroscopiche: il tempo richiesto per il legame del TF a uno dei due operatori differisce solo nel numero di tentativi di associazione macroscopica M, da comparare a quelli nel caso di un singolo operatore M_0. Di conseguenza, il rapporto tra il rate di associazione nel caso di due operatori e quello nel caso di un singolo operatore è dato dalla formula esatta $\frac{k}{k_0} = \frac{M_0}{M}$.

L'algoritmo consiste nei seguenti passaggi: inizializzazione di s^2, N (numero di coppie di basi del DNA) e M = 0; **associazione macroscopica**, in cui un numero random $\xi \in (1, N)$ è generato e la posizione del TF sul DNA è fissata a $\mathcal{P} = \xi$; **sliding**, in cui un numero random x è generato dalla distribuzione uniforme nell'intervallo unitario e a seconda di x uno dei seguenti eventi è scelto: (i) la posizione è cambiata a $\mathcal{P} + 1$ con probabilità $P_+ = s^2/(1 + 2s^2)$, (ii) la posizione è cambiata a $\mathcal{P} - 1$ con probabilità $P_- = s^2/(1 + 2s^2)$, o (iii) con probabilità $P_d = 1/(1 + 2s^2)$ **dissociazione microscopica** e **hopping**.

In questo caso la distanza di escursione r è generata dalla distribuzione di probabilità

$$\frac{dP(r)}{dr} = \frac{\left[\ln\left(\frac{r_{start}}{r_{min}}\right)\right]}{r\left[\ln\left(\frac{r}{r_{min}}\right)\right]^2} \qquad (3)$$



Se $r \geq r_{max}$ allora avviene una **dissociazione macroscopica**, M assume valore $M + 1$, e l'algoritmo riprende dall'inizio; se $r < r_{max}$ la distanza di hopping è calcolata estraendo un numero di passi n dall'istogramma corrispondente a r, ed estraendo z tramite la distribuzione in eq.(2). Anche un angolo ϕ è estratto dall'istogramma corrispondente. È consentito all'angolo un margine di variabilità pari a $\Delta\phi = 36°$. Se la coppia (z, ϕ) segue il cammino di un'elica, con una certa probabilità di associazione microscopica il TF si lega al DNA, altrimenti viene ripetuto l'hopping. Nel momento in cui il TF si lega al DNA, se non è legato al suo sito di legame specifico, la procedura continua ripartendo dallo sliding. La procedura termina nel momento in cui $\mathcal{P}$ corrisponde alla posizione del sito di legame del TF.

3 Programma

3.1 Random Walker

È la classe che si occupa di simulare un random walk nel piano 2D perpendicolare all'asse del DNA.

Public member functions:

- *RandomWalker()*: costruttore di default, inizializza la posizione di partenza del random walk dell'oggetto in $x = y = 0$.
- *RandomWalker(double x, double y)*: inizializza la posizione iniziale consentendo di specificare le coordinate di partenza.
- *RandomWalker(const RandomWalker &source)*: costruttore di copia.
- $\sim$*RandomWalker()*: distruttore.

- *Getstart(double frstart)*: assegna alla coordinata x il valore passato come argomento e ad y il valore 0.
- *RandomStep(double rstep, TRandom* a)*: estrae casualmente in maniera uniforme un angolo e fa fare un passo di lunghezza r_{step} in quella direzione.
- *RandomWalk(double rstep, TRandom* a)*: estrae casualmente una delle quattro direzioni cardinali e fa fare un passo di lunghezza r_{step} in quella direzione (per un random walk su reticolo quadrato).
- *Distanza()*: calcola la distanza dell'oggetto dall'origine.
- *Angolo()*: calcola l'angolo formato tra la retta che passa per l'origine e per le coordinate dell'oggetto e l'asse X.
- *Stampaposizione()*: Stampa la posizione attuale dell'oggetto e quella al passo precedente.

3.2 Data

Il piano perpendicolare all'asse del DNA è suddiviso in intervalli di distanze concentrici in base a r_{step}: per ognuno di questi intervalli, questa classe genera un istogramma con il numero di passi dei random walk passati per le distanze appartenenti all'intervallo, e un istogramma contenente la variazione in angolo rispetto al punto di partenza (*angolo di rebinding*) compiuta dai random walk passati per le distanze appartenenti all'intervallo. I valori da inserire negli istogrammi vengono prima salvati in array di vector. Gli array di vector sono così costituiti: a ogni vector corrisponde uno degli intervalli di distanze, e gli elementi in ogni vector in un caso sono i passi totali effettuati da un random walk, nell'altro gli *angoli di rebinding*.

Public member functions:

- *Data(double rmin, double rstep, double rmax)*: costruttore.
- *Data(const Data &source)*: costruttore di copia.
- $\sim$*Data()*: distruttore.
- *Add(double dist, int nstep)*: inserisce nel vector relativo alla distanza nell'argomento il numero di passi totali del random walk.
- *AddAngles(double dist, double angle, int nstep)*: inserisce nel vector relativo alla distanza nell'argomento l'angolo rispetto all'asse X a cui è terminato il random walk.
- *Stampa()*: stampa la matrice dei vector delle distanze.
- *Histos()*: genera gli istogrammi delle distanze.

- *ShowHistos()*: visualizza su schermo gli istogrammi delle distanze.
- *HistosAngles()*: genera gli istogrammi degli angoli.
- *ShowHistosAngles()*: visualizza su schermo gli istogrammi degli angoli.
- *Casual(double r)*: estrae casualmente un numero di passi dall'istogramma corrispondente alla distanza passata come argomento.
- *CasualAngles(double r)*: estrae casualmente un valore di variazione dell'angolo dall'istogramma corrispondente alla distanza passata come argomento.

3.3 TFactor

Questa classe modellizza il fattore di trascrizione (*transcription factor*), e ha bisogno dell'implementazione delle classi Data e TRandom per funzionare dato che alcuni metodi hanno bisogno di istanze di queste classi.

Public member functions:

- *TFactor(TRandom* poi)*: costruttore di default.
- *TFactor(double rmin, double rmax, double rstart, int N, TRandom* poi)*: costruttore.
- *TFactor (const TFactor &source)*: costruttore di copia.
- $\sim$*TFactor()*: distruttore.
- *SetPosition(TRandom* poi)*: determina la posizione del TF lungo l'elica con probabilità uniforme.
- *Selectr (TRandom* poi)*: ritorna una distanza casuale secondo la distribuzione (3).
- *Move(TRandom* poi)*: fa muovere il fattore di trascrizione, tramite sliding o hopping.
- *Hopping (double r, int n , TRandom* poi, double angle)*: tratta il processo di hopping per il TF.
- *Search(TRandom* poi, Data& p)*: si occupa di far iniziare la ricerca dell'operatore, richiamando i vari metodi precedenti e trattando i risultati che essi restituiscono.
- *ChangeTolleranza(double tolleranza)*: modifica il data member che tratta la tolleranza dell'angolo per cui il fattore di trascrizione può legarsi al DNA.
- *ChangeDistanza(double distanza)*: cambia la distanza tra i due operatori a cui si può ricollegare il fattore di trascrizione. Nel caso la distanza sia nulla si è nel caso di un singolo operatore presente.
- *SetPositionOperator(TRandom* poi)*: determina la posizione dei due operatori.

3.4 Simulation

Nella funzione simulation() vengono effettuati un numero pari a *walks* di Random Walk. Per ogni traiettoria che non superi r_{max} vengono salvate le distanze raggiunte nel vettore provvisorio *distanze*. Tramite i metodi della classe **Data** vengono generati gli istogrammi coi dati ricavati dai Random Walk. È possibile mostrare gli istrogrammmi creati settando la variabile *debugg* a kTRUE.
Nella seconda parte della funzione viene creata un'istanza della classe TFactor, in cui il numero N delle basi del DNA è settato a 20000, come in [1]. Si considerano prima un solo operatore e poi due operatori, facendo variare la distanza tra questi. In ciascun caso si fanno un numero di iterazioni pari a j per trovare il numero medio di dissociazioni macroscopiche.
In base a questi risultati, si calcola poi il rapporto tra i rate ($\frac{M_0}{M}$) per ciascuna distanza, e il rispettivo errore. Si plottano infine i risultati ottenuti in funzione della distanza tra gli operatori.

4 Esecuzione

I risultati riportati nella sezione 5 fanno riferimento a una simulazione in cui la variabile *searches* è stata posta a 10^5. Una simulazione così impostata richiede circa 9h 30 min di tempo. È comunque possibile ottenere un buon accordo con i valori previsti anche ponendo *searches* a 10^4 (tempo di esecuzione di circa un'ora), o comunque, riducendo ulteriormente il valore di *searches* ad esempio a 1000, è possibile verificare il funzionamento del programma e osservare un andamento dei risultati che si avvicina a quello atteso. Si eseguano i seguenti comandi per avviare la simulazione:

1. .L Compilemyclass.C
2. compilemyclass();
3. simulation();

5 Risultati

Come detto nella sezione precedente, sono state effettuate 10^5 simulazioni per ottenere una media aritmetica di M, come in [1]. Si sono selezionati $r_{max} = 11$nm, $s^2 = 700\text{bp}^2$, $r_{start} = 6.5$nm e $r_{min} = 5.5$nm seguendo quanto fatto in [1].
Nella prima parte della simulazione vengono quindi prodotti gli istogrammi per il numero di passi n e per l'angolo di legame ϕ. Si riportano come esempio gli istogrammi relativi alle traiettorie che sono passate per distanze appartenenti a $(r_{start}, r_{start} + r_{step})$, nelle figure 1 e 2.

Figura 1

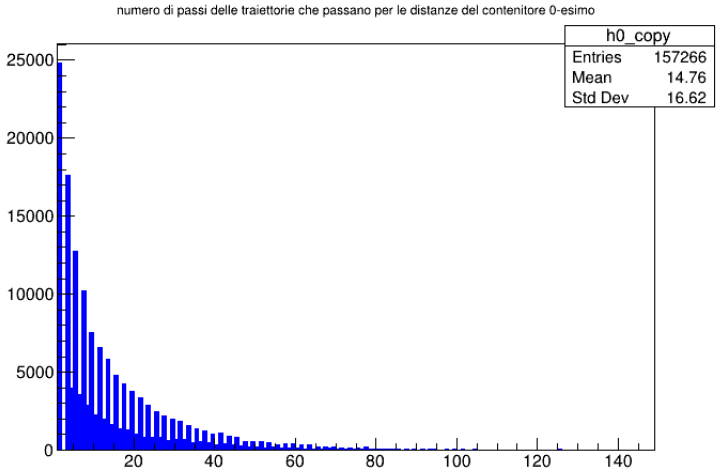
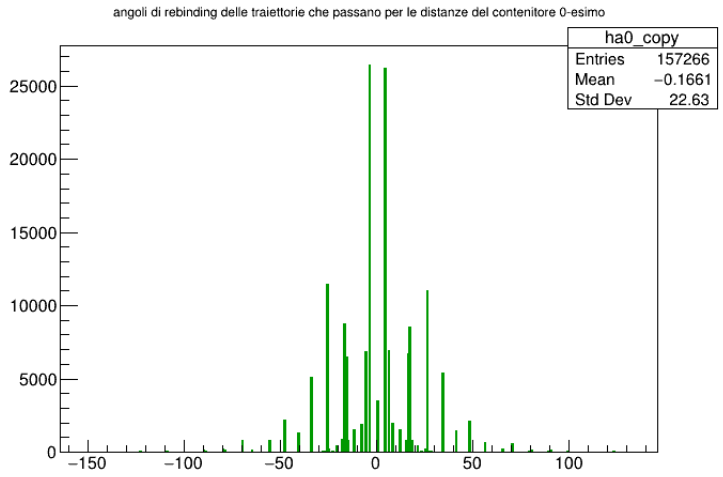
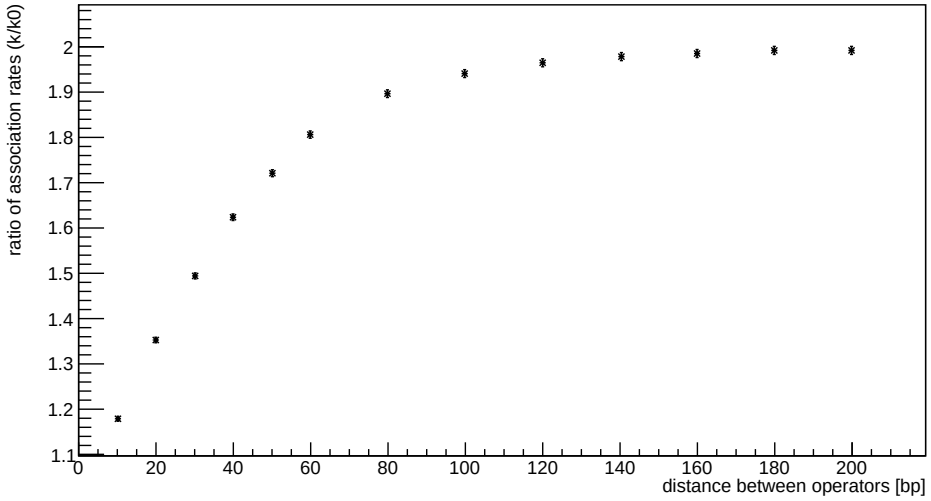


Figura 2



Infine, in figura 3 si riporta il plot dei rapporti dei rate da noi ricavati al variare delle distanze considerate, a confronto con quanto ottenuto dal fit dei dati sperimentali effettuato in [1], in figura 4.
Si può notare come il grafico da noi generato riproduca i risultati ottenuti in [1], con i nostri valori dei parametri per s^2 ed r_{max} (700 e 11), che sono in accordo con i dati sperimentali. Si nota come il rapporto tra i due rate sia prossimo a due quando la distanza tra gli operatori è maggiore di circa 65bp. Nel momento in cui questa distanza viene ridotta si nota invece come in entrambe le figure il rapporto tra i rate diminuisca e si avvicini a un valore unitario per una distanza tra gli operatori che tende allo zero.

Figura 3



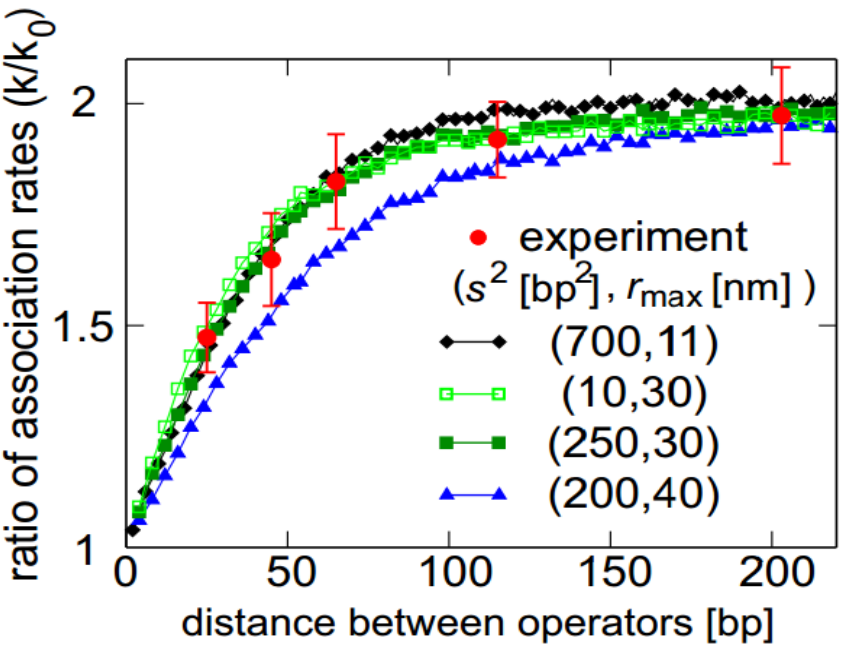


Figura 4

Riferimenti bibliografici

[1] K. Burzdy M. Tabaka and R. Holyst. *Physical Review E*, 92:022721–1, 2015.