Progetto Architetture 2024

Predizione struttura terziaria delle proteine

Fabrizio Angiulli, Fabio Fassetti, Simona Nisticò

a.a. 2024-25

1 Introduzione

Una proteina è una macromolecola biologica fondamentale per la vita. Rappresenta l'elemento costitutivo delle cellule e svolge una miriade di funzioni. Strutturalmente, le proteine sono costituite da lunghe catene di amminoacidi, uniti tra loro da legami peptidici. Esistono 20 amminoacidi diversi che, combinati in sequenze specifiche, danno origine a una grande varietà di proteine. Di una proteina si definiscono i seguenti quattro livelli di struttura.

Struttura primaria: sequenza lineare di amminoacidi.

Struttura secondaria: formazione di α-eliche e β-foglietti.

Struttura terziaria: ripiegamento 3D completo.

Struttura quaternaria: associazione di più catene proteiche.

La maggior parte degli sforzi della comunità bioinformatica si è concentrata sulla predizione della struttura terziaria in quanto:

- la struttura primaria è facilmente determinabile dal sequenziamento,
- la struttura secondaria è relativamente prevedibile dai pattern di sequenza,
- la struttura quaternaria spesso deriva dall'assemblaggio di strutture terziarie note.

Per quanto riguarda la struttura terziaria, invece, questa è cruciale per capire la funzione delle proteine però è difficile da determinare sperimentalmente in quanto le due tecniche di laboratorio principali ossia la cristallografia e la risonanza magnetica nucleare sono complesse e costose. La predizione della struttura terziaria rappresenta quindi il "collo di bottiglia" nella comprensione delle proteine. Quindi, la comunità scientifica si è adoperata per proporre soluzioni computazionali al problema della predizione della struttura o del ripiegamento (folding) delle proteine, ossia idi predire come la catena di amminoacidi si ripiega nello spazio per determinare la struttura 3D. La predizione della struttura tridimensionale della proteina si basa sul principio di parsimonia in accordo al quale la proteina tende naturalmente a raggiungere uno stato di minima energia rappresentante la conformazione più stabile. Di conseguenza le tecniche algoritmiche forniscono come predizione la struttura che minimizza l'energia.

2 Descrizione del problema

Formalmente, il problema che si vuole affrontare è di seguito definito.

Definizione 1 (Problema) *Data una sequenza di n aminoacidi, ossia una stringa di lunghezza n definita su un alfabeto di* 20 *caratteri, calcolare i vettori φ e ψ, entrambi di lunghezza n, che definiscono gli angoli formati dagli aminoacidi nello spazio tridimensionale, tali che la struttura 3D complessiva risulti a energia minima.*

Per giungere a una soluzione del problema bisogna affrontare due sfide, (i) data una struttura calcolarne l'energia, (ii) trovare la struttura a energia minima. Nella Sezione 3 vengono riportate le fuzioni per il calcolo dell'energia e nella Sezione 4 viene riportata la procedura di ricerca della soluzione a energia minima.

3 Calcolo dell'energia di una struttura

Il primo punto da affrontare è quindi quello di calcolare, data una struttura, la sua energia. Questa dipende da numerosi fattori, quali le forze di Van Der Waals, i legami idrogeno, le interazioni elettrostatiche, termini geometrici, ossia energia di legame e energia degli angoli diedri, potenziali statistici basati sulla conoscenza, termini di entropia conformazionale e l'effetto idrofobico, ossia la misura di quanto un aminoacido "non ami" l'acqua. In questo progetto consideriamo solo alcuni di questi fattori e in particolare:

1. l'*energia di Ramachandran*, ossia l'energia potenziale associata ai diversi angoli,
2. l'*energia idrofobica* basata sulle distanze 3D,
3. l'*energia elettrostatica* basata sulle distanze 3D,
4. l'*energia di impaccamento* basata sul volume e distanze 3D.

Per quanto riguarda la prima fonte di energia, questa si può calcolare direttamente a partire dai vettori φ e ψ, per le altre, invece, è necessario conoscere la posizione degli aminoacidi in termini di coordinate cartesiane. Pertanto, è necessario convertire le posizioni degli aminoacidi determinate dagli angoli in coordinate cartesiane. Questo può essere fatto con la seguente procedura che ricostruisce la backbone della catena a partire dalla sequenza e dai vettori degli angoli. Per posizionare correttamente gli aminoacidi dati gli angoli è necessario conoscere i principali atomi coinvolti e la distanza fisica tra di essi. La seguente figura riporta la struttura atomica di un aminoacido $\mathcal{AA}$ limitatamente agli atomi di principale interesse qui.

$$\begin{array}{c} \mathcal{AA} \\ | \\ \mathrm{N} - \mathrm{C}_\alpha - \mathrm{C} \end{array}$$

Le funzioni `backbone` e `rotation` consentono di ottenere le coordinate cartesiane a partire dagli angoli. Le funzioni `rama-energy`, `hydrophobic-energy`, `electrostatic-energy` e `packing-energy` riportano il calcolo delle associate componenti energetiche. La funzione `energy` illustra i passaggi per ottenere l'energia complessiva della struttura.

Function rotation($axis$,θ)

`// Matrice di rotazione base`
`// Restituisce la matrice di rotazione attorno a un asse` $axis$ `di un angolo theta` θ`, dove` $axis$ `è un vettore a 3 dimensioni`

1 $axis = axis/(axis \circ axis)$; `//` $axis \circ axis$ `è il prodotto scalare`
2 $a = \cos(\theta/2.0)$;
3 $[b, c, d] = -1 \cdot axis \cdot \sin(\theta/2.0)$;
4 **return**

$$\begin{bmatrix} a^2+b^2-c^2-d^2, & 2\cdot(b\cdot c+a\cdot d), & 2\cdot(b\cdot d-a\cdot c) \\ 2\cdot(b\cdot c-a\cdot d), & a^2+c^2-b^2-d^2, & 2\cdot(c\cdot d+a\cdot b) \\ 2\cdot(b\cdot d+a\cdot c), & 2\cdot(c\cdot d-a\cdot b), & a^2+d^2-b^2-c^2 \end{bmatrix}$$

Per il calcolo delle funzioni cos e sin si utilizzino le seguenti approssimazioni polinomiali:

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}, \qquad \sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}.$$

4 Algoritmo di ricerca della struttura a energia minima

Per quanto riguarda la ricerca della struttura a energia minima, essendo le possibili configurazioni tridimensionali un numero enorme, è necessario ricorrere a un algoritmo che calcoli soluzioni sub-ottime in tempi ragionevoli. Una delle tecniche di maggiore successo in questo contesto è il *simulated annealing*. Il simulated annealing ("ricottura simulata" in italiano) è una tecnica ispirata al processo metallurgico di ricottura. Tale processo, prevede

1. il riscaldamento del materiale metallico ad una temperatura elevata,
2. il mantenimento di questa temperatura per un determinato periodo di tempo,
3. il raffreddamento lento e controllato del materiale.

Questo trattamento consente di ridurre le tensioni interne del materiale aumentandone la duttilità e la lavorabilità. Chimicamente consente di modificare la struttura cristallina per ottimizzare determinate proprietà. La sua implementazione algoritmica segue fondamentalmente l'idea descritta attraverso i seguenti passi:

1. inizializzazione ad alta temperatura, ossia scelta di un valore molto alto per il parametro T,
2. campionamento Monte Carlo a T corrente
3. raffreddamento graduale con criterio di Metropolis per accettazione.

Il seguente algoritmo illustra in dettaglio i passi.

Function backbone(s, φ, ψ)

// Calcola le coordinate 3D approssimate della backbone usando gli angoli diedri φ e ψ

n = lunghezza di s;

// Distanze standard nel backbone (in Angstrom)

$r_{ca,n} = 1.46$; // Distanza CA-N

$r_{ca,c} = 1.52$; // Distanza CA-C

$r_{c,n} = 1.33$; // Distanza C-N

// Angoli standard del backbone (in radianti)

$\theta_{ca,c,n} = 2.028$;

$\theta_{c,n,ca} = 2.124$;

$\theta_{n,ca,c} = 1.940$;

// Crea la matrice coords con il risultato di dimensione $(n \cdot 3) \times 3$

// 3 coordinate (x, y, z), per i tre atomi N, C_α, C di ogni aminoacido

// Posiziona il primo aminoacido

$coords[0] = [0, 0, 0]$; // N

$coords[1] = [r_{ca,n}, 0, 0]$; // C_α

// Costruisce la catena

for i=0 **to** n **do**

  $idx = i \cdot 3$; // indice base per questo aminoacido

  if $i > 0$ **then**

    // Posiziona N usando l'ultimo C

    $v1 = coords[idx - 1] - coords[idx - 2]$; // vettore $C_\alpha - C$

    $v1 = v1/\|v1\|$;

    $rot = rotation(v1, \theta_{c,n,ca})$;

    $newv = [0, r_{c,n}, 0] \times rot$); // dove $\times$ indica il prodotto matriciale

    $coords[idx] = coords[idx - 1] + newv$;

    // Posiziona C_α usando φ

    $v2 = coords[idx] - coords[idx - 1]$;

    $v2 = v2/\|v2\|$;

    $rot = rotation(v2, \phi[i])$;

    $newv = [0, r_{ca,n}, 0] \times rot$; // dove $\times$ indica il prodotto matriciale

    $coords[idx + 1] = coords[idx] + newv$;

  // Posiziona C usando ψ

  $v3 = coords[idx + 1] - coords[idx]$;

  $v3 = v3/\|v3\|$;

  $rot = rotation(v3, \psi[i])$;

  $newv = [0, r_{ca,c}, 0] \times rot$); // dove $\times$ indica il prodotto matriciale

  $coords[idx + 2] = coords[idx + 1] + newv$;

return *coords*

Function rama-energy(φ, ψ)

// Energia di Ramachandran

n = lunghezza di φ (o, equivalentemente, di ψ);

$\alpha_\varphi, \alpha_\psi = -57.8, -47.0$;

$\beta_\varphi, \beta_\psi = -119.0, 113.0$;

$energy = 0$;

for $i = 0$ **to** n **do**

  $\alpha_{dist} = \sqrt{(\varphi[i] - \alpha_\varphi)^2 + (\psi[i] - \alpha_\psi)^2}$;

  $\beta_{dist} = \sqrt{(\varphi[i] - \beta_\varphi)^2 + (\psi[i] - \beta_\psi)^2}$;

  $energy = energy + 0.5 \cdot \min(\alpha_{dist}, \beta_{dist})$;

return *energy*

Function hydrophobic-energy(s, *coords*)

```
   // Energia idrofobica basata sulle distanze 3D
1  n = lunghezza di s;
2  ca_coords = coordinate degli atomi C_α;
3  energy = 0;
4  for i = 0 to n do
5      for j = i+1 to n do
           // Considera la distanza euclidea tra i C_α degli aminoacidi in posizione i e j
6          if dist(i, j) < 10.0 then
7              energy = energy + (hydrophobicity[s[i]] · hydrophobicity[s[j]])/dist(i, j)
8  return energy
```

Function electrostatic-energy(s, *coords*)

```
   // Energia elettrostatica basata sulle distanze 3D
1  n = lunghezza di s;
2  ca_coords = coordinate degli atomi C_α;
3  energy = 0;
4  for i = 0 to n do
5      for j = i+1 to n do
           // Se entrambi carichi
           // Considera la distanza euclidea tra i C_α degli aminoacidi in posizione i e j
6          if i ≠ j and dist(i, j) < 10.0 and charges[s[i]] ≠ 0 and charges[s[j]] ≠ 0 then
7              energy = energy + (charge[s[i]] · charge[s[j]])/(dist[i, j] · 4.0)
8  return energy
```

Function packing-energy(s, *coords*)

```
   // Energia di impaccamento basata sul volume e distanze 3D
1  n = lunghezza di s;
2  ca_coords = coordinate degli atomi C_α;
3  energy = 0;
4  for i = 0 to n do
5      density = 0;
6      for j = 0 to n do
           // Considera la distanza euclidea tra i C_α degli aminoacidi in posizione i e j
7          if i ≠ j and dist(i, j) < 10.0 then
8              density = density + volume[s[j]]/dist(i, j)^3
9      energy = energy + (volume[s[i]] − density)^2
10 return energy
```

Function energy(s, φ, ψ)

```
1  coords = backbone(s, φ, ψ);
   // Calcolo delle componenti energetiche
2  rama_e = rama-energy(φ, ψ);
3  hydro_e = hydrophobic-energy(s, coords);
4  elec_e = electrostatic-energy(s, coords);
5  pack_e = packing-energy(s, coords);
   // Pesi per i diversi contributi
6  w_rama = 1.0;
7  w_hydro = 0.5;
8  w_elec = 0.2;
9  w_pack = 0.3;
   // Energia totale
10 total_e = w_rama · rama_e + w_hydro · hydro_e + w_elec · elec_e + w_pack · pack_e;
11 return total_e
```

Algorithm 1: Simulated annealing

Input: sequenza aminoacidica s, temperatura iniziale T_0, tasso di raffreddamento α, costante k
Output: vettori di angoli φ e ψ

```
1  begin
2      n = lunghezza di s;
3      T = T_0;
       // genera una soluzione casuale iniziale
4      φ = vettore casuale di n numeri reali tra −π e π;
5      ψ = vettore casuale di n numeri reali tra −π e π;
6      E = energy(s, φ, ψ);
7      t = 0;
8      repeat
           // genera un vicino della soluzione corrente, perturbando un angolo
9          i = posizione casuale tra 0 e n;
10         ϑ_φ = valore reale tra −π e π;
11         φ[i] = φ[i] + ϑ_φ;
12         ϑ_ψ = valore reale tra −π e π;
13         ψ[i] = ψ[i] + ϑ_ψ;
           // calcola la variazione di energia della nuova configurazione
14         ΔE = energy(s, φ, ψ) − E;
15         if ΔE ≤ 0 then
16             accetta la nuova configurazione;
17             E = energy(s, φ, ψ);
18         else
               // calcola probabilità di accettazione
19             P = e^{−ΔE/(kT)};
20             r = numero random tra 0 e 1;
21             if r ≤ P then
22                 accetta la nuova configurazione;
23                 E = energy(s, φ, ψ);
24             else
25                 rifiuta la nuova configurazione;
                   // ripristina φ e ψ precedenti
26                 φ[i] = φ[i] − ϑ_φ;
27                 ψ[i] = ψ[i] + ϑ_ψ;
           // aggiorna la temperatura
28         t = t + 1;
29         T = T_0 − √(α · t);
30     until T ≤ 0;
31 return φ, ψ
```

5 Descrizione dell'attività progettuale.

Obiettivo del progetto è mettere a punto un'implementazione dell'algoritmo di predizione della struttura terziaria delle proteine in linguaggio C e di migliorarne le prestazioni utilizzando le tecniche di ottimizzazione basate sull'organizzazione dell'hardware.

L'ambiente sw/hw di riferimento è costituito dal linguaggio di programmazione C (`gcc`), dal linguaggio assembly x86-32+SSE e dalla sua estensione x86-32+AVX (`nasm`) e dal sistema operativo Linux (`ubuntu`).

In particolare il codice deve consentire di trovare i vettori degli angoli φ e ψ tramite l'algoritmo precedentemente illustrato, considerando i valori dei parametri: `-to <`T_0 `temperatura iniziale>`, `-alpha <`α `tasso di raffredamento>`, `-k <`k `costante>` e `-sd <seed per la generazione casuale>`.

Quindi la chiamata avrà la seguente struttura:

```
./pst<arch> -seq <SEQ> -to <t0> -alpha <alpha> -k <k> -sd <sd>
    <arch>: architettura associata all'eseguibile, {32c, 64c, 32ompc 64ompc}
    <SEQ>: file ds2 la sequenza
    <t0>: temperatura iniziale
    <alpha>: tasso di raffredamento
    <k>: costante
    <sd>: seed per la generazione casuale
```

e sorgenti ed eseguibili devono essere contenuti in una cartella il cui nome è l'id del gruppo. Qualora un valore di un parametro (sia esso di default o specificato dall'utente) non sia applicabile, il codice deve segnalarlo con un messaggio e terminare.

Di seguito si riportano ulteriori linee guida per lo svolgimento del progetto:

- Si consiglia di affrontare il progetto nel seguente modo:

  1. Codificare l'algoritmo interamente in linguaggio C, possibilmente come sequenza di chiamate a funzioni;
  2. Sostituire le funzioni scritte in linguaggio ad alto livello che necessitano di essere ottimizzate con corrispondenti funzioni scritte in linguaggo assembly.

  Ciò consentirà di verificare che l'algoritmo che si intende ottimizzare è corretto e di gestire più facilmente la complessità del progetto.

- Al fine di migliorare la valutazione dell'attività progettuale è preferibile presentare nella relazione un confronto tra le prestazioni delle versioni intermedie, ognuna delle quali introduce una particolare ottimizzazione, e finale del codice. Obiettivo del confronto è sostanziare la bontà delle ottimizzazioni effettuate.

- Occorre lavorare in autonomia e non collaborare con gli altri gruppi. Soluzioni troppo simili riceveranno una valutazione negativa. I progetti verranno messi in competizione.

- Sono richieste due soluzioni software, la prima per l'architettura x86-32+SSE e la seconda per l'architettura x86-64+AVX.

  Per i dettagli riguardanti la redazione del codice fare riferimento al file sorgente *c* `pst32c.c`, al file sorgente *nasm* `pst32.nasm`, allo script eseguibile `runpst32` (versione `x86-32+SSE`) e al file sorgente *c* `pst64c.c`, al sorgente *nasm* `pst64.nasm`, allo script eseguibile `runpst64` per la versione `x86-64+AVX` disponibili sulla piattaforma.

  Per l'interfacciamento tra programmi in linguaggio C e programmi in linguaggio assembly fare riferimento al documento allegato alla descrizione del progetto.

- È inoltre richiesta per la soluzione 64-AVX, una versione che faccia uso delle istruzioni `OpenMP`. I nomi dei relativi file dovranno contenere il suffisso "`_omp`" (es. `pst64_omp.c`).

- Il software dovrà essere corredato da una relazione. Per la presentazione del progetto è possibile avvalersi di slide.

- Prima della data di consegna del progetto verranno pubblicate le convenzioni da rispettare riguardanti i nomi e la collocazione dei file/directory al fine della compilazione e l'esecuzione dei codici di programma mediante script appositamente predisposti. **Dato l'elevato numero di progetti, sarà cura del candidato accertarsi di aver rispettato pienamente le convenzioni di consegna.**

Buon lavoro!