UNIVERSITÀ DEGLI STUDI DI ROMA TRE
FACOLTÀ DI SCIENZE M.F.N.

Confronto tra sequenze in bioinformatica

Sintesi della tesi di Laurea in Matematica

di Raffaella Fanuli
Relatore: Pasquale Caianiello

Negli ultimi anni si è assistito allo sviluppo di una nuova area di ricerca nota come Biologia Computazionale (o Bioinformatica), la quale è una disciplina che si occupa di una molteplicitá di problemi diversi, tutti peró originatisi nel campo della Biologia Molecolare.

La nascita della Biologia Computazionale puó datarsi intorno al 1990, anno in cui il Department of Energy (DOE), l' Office of Health and Enviromental Research e il National Institute on Health (NIH) degli Stati Uniti hanno dato il via ad un progetto mondiale noto come Human Genome Proiect. L'obiettivo finale, era la lettura dell'intero genoma umano nonché di altre specie - animali e non - e, in ultima analisi, la sua interpretazione al fine di dare vita a una nuova medicina molecolare.

Il tema centrale della Biologia Computazionale è la computazione su sequenze molecolari (stringhe); questo costituisce un importante punto di contatto tra la Biologia e l'Informatica in quanto la computazione su stringhe è materia di ricerca di notevole interesse in diversi settori di quest'ultima. Una branca, fondamentale in questo ambito, è lo studio di algoritmi su stringhe. In passato sono stati sviluppati diversi algoritmi su sequenze, in particolare

molti di questi rientrano in un settore noto come pattern matching comprendente differenti problemi che derivano dalla problematica generale di ricercare particolari stringhe pattern in un testo.
Un altro settore strettamente collegato riguarda il confronto tra stringhe (o sequenze) e comprende problemi, quali la distanza di editing tra stringhe, la piú lunga sottosequenza comune e la piú corta supersequenza comune.

Una fondamentale assunzione in Biologia Molecolare è che si possano ottenere risultati biologici significativi considerando il DNA come una stringa monodimensionale (su un alfabeto di 4 simboli) astraendo dalla reale natura del DNA quale molecola tridimensionale. Una tale assunzione è stata fatta anche per le proteine, in quanto la conformazione dell'avvolgimento tri-dimensionale delle proteine è completamente determinato nella sequenza lineare codificante la proteina (in questo caso l'alfabeto è di 20 simboli). Pertanto la Biologia Molecolare riguarda sequenze: essa riduce fenomeni biochimici complessi alla interazione tra sequenze definite. Non sorprende allora il fatto che problemi fondamentali in Biologia Molecolare siano definiti come problemi computazionali su sequenze. Nuovi algoritmi su stringhe trovano quindi applicazione diretta in Biologia Molecolare, mentre problemi nuovi su stringhe emergono da questo contesto, richiedendo spesso nuove tecniche algoritmiche per la loro soluzione.

Gli strumenti di questa nuova scienza sono dunque le banche dati delle biosequenze (ne esistono decine) e tutti quei softwares che consentono di analizzarle, confrontarle ecc.
Questi strumenti sono resi accessibili a tutti attraverso Internet che offre una vastissima gamma di siti che permette agli utenti di effettuare analisi e confronti immediati su sequenze di loro interesse.
Scopo di questa tesi è di illustrare gli algoritmi e gli strumenti che costituiscono le fondamenta concettuali dell'universo di soluzioni proposte dalla rete al problema del confronto fra sequenze biologiche, cercando di mantenere un riferimento al concreto problema biologico che le soluzioni prospettate possono risolvere.
Presentiamo una sintesi dei capitoli che costituiscono la tesi.

1. Definizioni di base

Definizione 1. *Un* **alfabeto** $\mathcal{A}$ *è un insieme finito di elementi detti simboli, caratteri o lettere.*
Sia $\mathcal{A}^n$ *l'insieme di tutte le parole su* $\mathcal{A}$ *con lunghezza pari ad* n*, si ha che* $\mathcal{A}^* = \bigcup_{n=0}^{\infty} \mathcal{A}^n$ *è l'insieme di tutte le parole finite su* $\mathcal{A}$*.*

Definizione 2. *Una* **stringa** *o* **parola s** *sull'alfabeto* $\mathcal{A}$ *è una sequenza finita o nulla di simboli di* $\mathcal{A}$*:*

$$\mathbf{s} = (a_1, a_2, ..., a_n), a_i \in \mathcal{A} \tag{1}$$

Definizione 3. *Una parola* $\mathbf{s_1}$ *è un* **fattore sinistro** *o un* **prefisso** *di una parola* $\mathbf{s} \in \mathcal{A}^*$ *se esiste una parola* $\mathbf{s_2} \in \mathcal{A}^*$ *tale che* $\mathbf{s} = \mathbf{s_1 s_2}$*.*
Indichiamo con $\mathbf{s_{[0,i]}}$ *il prefisso di* $\mathbf{s}$ *composto dalle sue prime i lettere.*

2. Confronto fra sequenze

Confrontare tra loro due sequenze proteiche ha come scopo quello di trovarne delle similarità.
L'idea generale è che le sequenze aminoacidiche determinano la forma della proteina e questa ne determina la funzione biologica. Dunque, studiando la similarità tra le sequenze, noi cerchiamo di scoprire o di confermare se ci sono delle analogie di forma e funzione. Questo approccio si è rivelato spesso di successo.

Per quantificare la similarità possono essere utilizzate due grandezze: la misura di similarità o considerare una *distanza*.
La misura di similarità associa valori più grandi a sequenze più similari, mentre la distanza vi associa valori più piccoli. In molti casi distanza e misura di similarità sono intercambiabili: **provare a minimizzare la distanza tra sequenze è la stessa cosa che provare a massimizzarne la similarità**.

In generale, le distanze soddisfano gli assiomi di una metrica.

Per poter effettuare il confronto tra sequenze vengono introdotti i concetti di :

- *Operazioni di editing*: sono le operazioni primitive svolte per "correggere" le differenze tra le sequenze.
Si hanno le seguenti operazioni di editing: **substitution (a,b)**, cambio di un carattere in un altro; **deletion (a,-)**, cancellazione di un carattere; **insertion (-,b)** , inserimento di un carattere.
Ad ognuna di esse è associato un valore reale non negativo tramite una **funzione costo** δ.

- *Allineamento tra sequenze* : è la successione di operazioni di editing necessarie per trasformare una sequenza in un'altra.

- *Costo dell'allineamento* : somma dei costi di tutte operazioni di editing che compongono l'allineamento.

- *Allineamento ottimale*: è un allineamento che ha il minimo costo tra tutti gli allineamenti possibili.

- *Distanza di editing* tra le sequenze $\mathbf{x}$ e $\mathbf{y}$: è il costo di un allineamento ottimale di $\mathbf{x}$ e $\mathbf{y}$ rispetto alla funzione costo δ e si indica con $d_\delta(\mathbf{x}, \mathbf{y})$.

Nell'ambito di questo discorso, vengono presentati alcuni algoritmi noti in letteratura basati sulla tecnica della programmazione dinamica:

- PROGRAMMAZIONE DINAMICA che calcola la distanza di editing tra due sequenze $\mathbf{s}$ e $\mathbf{t}$ (di lunghezza rispettivamente pari a m e n)

  Le linee base del procedimento sono le seguenti:
  Consideriamo due prefissi $\mathbf{s_{[0,i]}}$ e $\mathbf{t_{[0,j]}}$, con $i, j \geqslant 1$. Assumiamo di conoscere già gli allineamenti ottimali tra tutti i prefissi di lunghezza inferiore a quelle di $\mathbf{s_{[0,i]}}$ e $\mathbf{t_{[0,j]}}$, in particolare di

  1. $\mathbf{s_{[0,i-1]}}$ e $\mathbf{t_{[0,j-1]}}$
  2. $\mathbf{s_{[0,i-1]}}$ e $\mathbf{t_{[0,j]}}$
  3. $\mathbf{s_{[0,i]}}$ e $\mathbf{t_{[0,j-1]}}$.

Un allineamento ottimale di $\mathbf{s}_{[\mathbf{0},\mathbf{i}]}$ e $\mathbf{t}_{[\mathbf{0},\mathbf{j}]}$ si otterrà come un'estensione di uno di questi allineamenti per mezzo di una delle seguenti operazioni:

1. una *substitution* ($\mathbf{s_i}$,$\mathbf{t_j}$), o un *match* ($\mathbf{s_i}$,$\mathbf{t_j}$) se $\mathbf{s_i} = \mathbf{t_j}$,
2. una *deletion* ($\mathbf{s_i}, -$), o
3. una *insertion* ($-, \mathbf{t_j}$).

Si sceglierà allora semplicemente :

$$\begin{aligned} d_\delta(\mathbf{s}_{[\mathbf{0},\mathbf{i}]}, \mathbf{t}_{[\mathbf{0},\mathbf{j}]}) = min\{&d_\delta(\mathbf{s}_{[\mathbf{0},\mathbf{i-1}]}, \mathbf{t}_{[\mathbf{0},\mathbf{j-1}]}) + \delta(\mathbf{s_i}, \mathbf{t_j}), \\ &d_\delta(\mathbf{s}_{[\mathbf{0},\mathbf{i-1}]}, \mathbf{t}_{[\mathbf{0},\mathbf{j}]}) + \delta(\mathbf{s_i}, -), \\ &d_\delta(\mathbf{s}_{[\mathbf{0},\mathbf{i}]}, \mathbf{t}_{[\mathbf{0},\mathbf{j-1}]}) + \delta(-, \mathbf{t_j})\} \end{aligned} \tag{2}$$

E, per un dato δ, le distanze di editing di tutti i prefissi di $\mathbf{s}$ e $\mathbf{t}$ definiranno una **matrice distanza** $D = (d_{i,j})$, di dimensione $(m+1) \times (n+1)$, con coefficienti $D[i,j] = d_{i,j} = d_\delta(\mathbf{s}_{[\mathbf{0},\mathbf{i}]}, \mathbf{t}_{[\mathbf{0},\mathbf{j}]})$.
L'ultimo coefficiente della matrice D, dunque, conterrà il risultato desiderato $d_\delta(\mathbf{s}, \mathbf{t})$ e tutti i percorsi da $(0,0)$ a (m,n) rappresentano allineamenti di $\mathbf{s}$ e $\mathbf{t}$.

- ALLINEAMENTO che costruisce un allineamento ottimale procedendo a ritroso nella procedura di PROGRAMMAZIONE DINAMICA. Più precisamente risale la matrice dalla posizione $D[m,n]$ alla $D[0,0]$ prendendo ad ogni passaggio la giusta direzione.

- ALLINEAMENTI OTTIMALI che calcola tutti gli allineamenti ottimali possibili tra due sequenze sempre andando a ritroso nella matrice D.

Si ha il seguente risultato:

Proposizione 1. *L'algoritmo PROGRAMMAZIONE DINAMICA possiede una complessità temporale pari a* $\mathcal{O}(mn)$.
La complessità spaziale è pari a:
- $\mathcal{O}(min(m,n))$ *se viene richiesto solo il valore della distanza di editing tra* $\mathbf{s}$ *e* $\mathbf{t}$*;*
- $\mathcal{O}(mn)$ *se viene richiesta anche la ricostruzione degli allineamenti ottimali.*

Altri algoritmi riportati, che risolvono variazioni del problema originale e che sono delle variazioni degli algoritmi precedenti sono: ALLINEAMENTO LOCALE, ALLINEAMENTO LOCALE K DIFF, GAP MODEL.

3. Alberi filogenetici

Tutte le discipline biologiche sono unite dall'idea che tutti gli organismi viventi condividano una storia comune.
La storia genealogica della vita è di solito rappresentata utilizzando la struttura matematica degli alberi che , in questo caso, sono detti **alberi filogenetici**. Un **albero filogenetico** è un albero utilizzato per modellizzare la "storia evolutiva" di un gruppo di sequenze o di organismi di interesse, dove, le sue foglie sono etichettate con le specie o le sequenze note che si vogliono confrontare e i nodi interni rappresentano ipotetici predecessori **incogniti** degli oggetti iniziali.

Quando tutti i rami di un albero sono etichettati con valori positivi si parla di **alberi additivi**. Negli alberi filogenetici additivi tali valori indicano in generale l'ammontare dei cambiamenti evolutivi avvenuti nel passaggio da un nodo al suo predecessore.
Di particolare importanza dal punto di vista evolutivo sono un tipo particolare di alberi additivi : gli **alberi ultrametrici**. Questi sono alberi tali che, per ogni sottoalbero, le foglie sono equidistanti dalla radice.

Una categoria di metodi per la costruzione di alberi si basa sull' osservazione che **gli alberi stessi possono essere rappresentati dalle distanze**. Tali metodi sono detti $\mathbf{metodi - distanza}$ e cercano di convertire la distanza tra due sequenze in alberi filogenetici.
Alla base di tali metodi c'è la supposizione che, secondo un qualche criterio biologico, si sia associata ad un insieme di sequenze $\mathbf{S_1}, \mathbf{S_2}, ..., \mathbf{S_n}$ un'attendibile distanza $d(\mathbf{S_i}, \mathbf{S_j}) = d_{i,j}$ tale che
- $d_{i,j} \geq 0$;
- $d_{i,j} = d_{j,i}$.

La matrice D i cui coefficienti sono le distanze $d_{i,j}$ è detta **matrice distanza**

e, per come è definita d, essa è una matrice simmetrica con valori positivi fuori dalla diagonale e pari a zero sulla diagonale.

Una tecnica per la costruzione di alberi, che impiega le distanze, consiste nell'utilizzare generali metodi di *clustering*.

Definizione 4. *Un clustering di un singolo livello su un insieme di oggetti M è semplicemente una partizione di M i cui elementi sono detti clusters (grappoli).*
Un clustering gerarchico di M è una collezione $\mathfrak{H}$ di sottoinsiemi di M tale che:
$\{m\} \in \mathfrak{H}\ \forall\ m \in M$
$M \in \mathfrak{H}$
$\forall\ C, D \in \mathfrak{H}, C \cap D = 0,\ o\ \ C \subset D\ o\ D \subset C.$

Un clustering gerarchico $\mathfrak{H}$ può essere visto come un albero che consiste negli elementi di $\mathfrak{H}$ ordinati per inclusione, dove M è la radice e i singoli elementi $\{m\}$ sono le foglie.

Un algoritmo euristico che rappresenta un'applicazione particolare dei metodi di clustering è l'UPGMA (Unweighted Pair Group Method with Aritmetic) dovuto a Sneath e Sokej ([SS]) .
UPGMA permette di costruire un albero ultrametrico partendo da una matrice distanza ultrametrica.

Definizione 5. *Una matrice distanza D si definisce* **matrice ultrametrica** *se, comunque presi tre suoi coefficienti, si ha che due di essi hanno stesso valore e il terzo è minore uguale dei primi due.*

Riportiamo i passi dell'algoritmo UPGMA :
UPGMA
INPUT: Una matrice ultrametrica D di dimensione $(n \times n)$
OUTPUT: Albero ultrametrico T.
Inizializzazione

- Vengono inizializzati n clusters con gli oggetti dati ponendo un oggetto per cluster
- Viene posta la cardinalità di ogni cluster pari ad 1 : $n_i = 1 \; \forall i = 1 \dots n$
- Nell'albero di output viene assegnata una foglia per ogni oggetto

Iterazione

1. Vengono cercati i valori i e j tali che $D_{i,j}$ sia minimale e si crea un nuovo cluster (ij) che possiede $n_{ij} = n_i + n_j$ elementi.

2. Si connettono i e j ad un nuovo nodo sull'albero di output corrispondente al nuovo cluster (ij) e si assegna a ciascuno dei due rami che collegano le posizioni i e j al cluster (ij) una lunghezza pari a $\frac{D_{i,j}}{2}$.

3. Viene calcolata la distanza tra il nuovo cluster e tutti gli altri (tranne i e j) attraverso una media pesata delle distanze dai suoi componenti:

$$D_{(i,j),k} = \frac{n_i}{n_i + n_j} D_{i,k} + \frac{n_j}{n_i + n_j} D_{j,k} \tag{3}$$

4. Si eliminano le colonne e le righe di D che corrispondono ai clusters i e j e si aggiunge una riga e una colonna per il cluster (ij), con $D_{(ij),k}$ calcolato come nel punto 3.

5. Si ritorna al punto 1 finché non rimane un solo cluster.

La complessità temporale e quella spaziale di questo algoritmo sono pari a $\mathcal{O}(n^2)$.

Un altro algoritmo euristico illustrato, che invece porta alla costruzione di un albero additivo senza radice, è il Neighbor Joining (NJ), proposto da Saitou e Nei nel 1987 ([SN]),

Consideriamo un insieme di sequenze $\mathbf{S_1}, \dots, \mathbf{S_n}$ e sia D la matrice distanza relativa ad una distanza d . L'idea base di questo algoritmo è quella di confrontare tra loro coppie di foglie e di costruire dei clusters che però non siano contenuti l'uno nell'altro, ma che anzi siano "separati" dal resto. Per far ciò i clusters costruiti da NJ sono formati dai cosiddetti *vicini* (neighbors) che sono definiti come due foglie che sono in relazione tra loro più che con tutte le altre e che per questo sono connesse attraverso un solo nodo all'

albero.
Lo scopo dell'algoritmo è quello di minimizzare la lunghezza di ogni ramo, (*minimum evolution criterion*).
La complessità di NJ è la stessa di UPGMA.

4. Allineamenti multipli

In questo capitolo si affronta il problema di allineare tra loro più di due sequenze. Un'importante motivazione dal punto di vista biologico di tale studio, è il fatto che i databases catalogano le proteine in *famiglie proteiche*, dove per famiglia proteica si intende una collezione di proteine aventi struttura o funzione o storia evolutiva simile.
Può accadere, ad esempio, che una nuova sequenza proteica, sottoposta al confronto con quelle del database, non abbia particolari somiglianze con nessuna singola sequenza considerata, ma che invece possegga forti similarità con una famiglia proteica.
Introduciamo il concetto di *allineamento multiplo*

Definizione 6. *Sia $\mathcal{A}$ un alfabeto tale che $\{-\} \notin \mathcal{A}$ e sia $\mathcal{A}' = \mathcal{A} \cup \{-\}$. Date n sequenze $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n} \in \mathcal{A}^*$ di lunghezza k_1, k_2, ..., k_n, con $n > 2$, un* **allineamento multiplo delle sequenze $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n}$** *è una matrice $n \times l$*

$$\mathcal{MA} = \begin{pmatrix} \mathbf{S'_1} \\ \mathbf{S'_2} \\ . \\ . \\ . \\ \mathbf{S'_n} \end{pmatrix} = \begin{pmatrix} s'_{1,1} & s'_{1,2} & \ldots & \ldots & s'_{1,l} \\ s'_{2,1} & s'_{2,2} & \ldots & \ldots & s'_{2,l} \\ . & & & & \\ . & & & & \\ . & & & & \\ s'_{n,1} & s'_{n,2} & \ldots & \ldots & s'_{n,l} \end{pmatrix} \tag{4}$$

tale che:

1. $\mathcal{MA}[i][j] \in \mathcal{A}' \quad \forall \ 1 \leqslant i \leqslant n, \ 1 \leqslant j \leqslant l;$
2. $l = |\mathbf{S'_1}| = |\mathbf{S'_2}| = \cdots = |\mathbf{S'_n}| = |\mathcal{MA}|$ *con* $Max\{k_1, \ldots, k_n\} \leqslant l \leqslant \sum_{i=1}^{n} k_i;$

3. togliendo tutti i gaps da $\mathbf{S'_i}$ *si ottiene* $\mathbf{S_i}$, $\forall\, 1 \leqslant i \leqslant n$.

Per il costo di un allineamento multiplo, sono presenti delle controversie relative alla migliore definizione per tale valore, dando vita a diversi metodi di *scoring*.
Eccone alcuni:
- il *sum of pairs* , che è il più diffuso;
- Calcolo della distanza da una "*Sequenza Consenso*";
- Calcolo dei costi degli alberi filogenetici relativi alle sequenze.

Ad esempio, il sum of pairs è così definito:

Definizione 7. *Sia* δ *una funzione costo su* $\mathcal{A}'$ *che soddisfi gli assiomi di una metrica, e sia* $\mathcal{MA}$ *un allineamento multiplo delle sequenze* $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n} \in \mathcal{A}^*$, *Il costo* **sum of pairs (SP)** *di* $\mathcal{MA}$ *è dato dalla somma dei costi* δ *degli* $\binom{n}{2}$ *allineamenti indotti da* $\mathcal{MA}$, *su tutte le coppie possibili tra le n stringhe considerate.*

$$SP(\mathcal{MA}) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \delta(\mathbf{S'_i}, \mathbf{S'_j}) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{|\mathcal{MA}|} \delta(s'_{i,k}, s'_{j,k}) \tag{5}$$

Un algoritmo per costruire gli allineamenti multipli consiste nel generalizzare l'agoritmo PROGRAMMAZIONE DINAMICA sviluppato per l'allineamento tra due sequenze. In questo caso, invece di una matrice a due dimensioni, si avrà una tavola n-dimensionale MD, dove n è il numero di sequenze considerate e l'elemento $MD[i_1, i_2, \ldots, i_n]$ contiene il costo dell'allineamento ottimale tra i prefissi di lunghezza $i_1, i_2, \ldots, i_n$ rispettivamente delle sequenze $\mathbf{S_1}, \mathbf{S_2}$..., $\mathbf{S_n}$ rispetto ad una funzione ρ tale che

$$\rho : \overbrace{\mathcal{A}' \times \mathcal{A}' \times \cdots \times \mathcal{A}'}^{n} \longrightarrow \mathbb{R}^+ \tag{6}$$

.
Si ottiene che il calcolo in programmazione dinamica dell'allineamento mul-

tiplo ρ -ottimale di n sequenze, si basa sulla seguente formula ricorsiva:

$$MD[i_1, i_2, \ldots, i_n] = \\ min_{\varepsilon \in \{0,1\}^n, \varepsilon \neq (0,\ldots,0)} \{MD[(i_1, i_2, \ldots, i_n) - \varepsilon] + \rho(c_1, c_2, \ldots, c_n)\} \quad (7)$$

Con $MD(0, 0, \ldots, 0) = 0$ e

$$c_j = \begin{cases} \mathbf{S_{j i_j}} & \text{se } \varepsilon_j = 1; \\ - & \text{altrimenti.} \end{cases} \qquad 1 \leqslant j \leqslant n \quad (8)$$

Il valore per eccellenza di ρ è la funzione *sum of pairs* che porta all'allineamento SP-ottimale.

Tale algoritmo però, ha complessità temporale talmente elevata che nella realtà è applicabile solo per allineamenti multipli di 3, 4 sequenze. Dunque anche se questo algoritmo esiste, sfortunatamente nella realtà è poco utilizzabile.

L'algoritmo ideale sarebbe un algoritmo che allinei centinaia di sequenze e che abbia una complessità temporale polinomiale sia rispetto al numero delle sequenze, sia rispetto alla loro lunghezza . Sfortunatamente, è molto arduo riuscire a costruire un tale algoritmo, si ha infatti il seguente risultato ottenuto da Wang e Jiang ([WJ])

Teorema 1. *Il problema della ricerca dell'allineamento SP ottimale è un problema NP-completo.*

Però, anche se ricercare un algoritmo più efficiente per il caso generale vorrebbe dire risolvere un problema NP-completo, per alcuni casi particolari esistono diverse tecniche che possono essere applicate facilmente.
Una di queste è la cosiddetta **tecnica di Carrillo-Lipman** ([CL]) messa a punto per la ricerca di allineamenti multipli SP-ottimali e che rende più veloce l'algoritmo in programmazione dinamica.
Questa tecnica è tale da migliorare in modo significativo la complessità temporale, però, al fine di ottenere tale diminuzione, i dati di input devono

soddisfare le seguenti condizioni:
- occorre considerare un piccolo numero di sequenze;
- le sequenze considerate devono avere tra loro un'alta similarità.
Fortunatamente, la maggior parte degli input dati nelle ricerche biologiche posseggono queste due caratteristiche.
Il metodo di Carrillo-Lipman è basato sull'osservazione che se due sequenze sono particolarmente simili, allora il percorso sulla matrice MD sarà principalmente concentrato sulla diagonale, dunque non tutti i valori della tavola multidimensionale necessiteranno di essere calcolati.
Più precisamente, ognuna delle $\binom{n}{2}$ "facce" della tavola n-dimensionale definisce una matrice distanza tra le due sequenze corrispondenti. La proiezione dell'allineamento multiplo $\mathcal{MA}$ su ciascuna delle facce definisce un allineamento tra le due sequenze. Lo scopo è trovare gli estremi delle proiezioni del possibile allineamento ottimale e quindi contornare un'area interna alla tavola nella quale ricercare la soluzione.
Riassumiamo questo metodo:

1. Per ogni coppia di sequenze $\mathbf{S_i}$,$\mathbf{S_j}$ si calcola la distanza di editing $D(\mathbf{S_i}, \mathbf{S_j}) = d_\delta(\mathbf{S_i}, \mathbf{S_j})$.

2. Si calcola per ogni piano (u, v) della tavola multidimensionale MD **l'estremo di Carrillo-Lipman**

$$U - \sum_{i<j,\ (i,j)\neq(u,v)} D(\mathbf{S_i}, \mathbf{S_j}) \tag{9}$$

   Dove con U si indica un estremo superiore noto del costo dell'allineamento SP-ottimale $\mathcal{MA}^*$

3. Per ogni piano (u, v) si ricercano tutte le posizioni (i, j) nel piano tali che il costo del miglior cammino possibile che passa per (i, j)

$$D(\mathbf{S_{u[1,i-1]}}, \mathbf{S_{v[1,j-1]}}) + \delta(\mathbf{S_{u_i}}, \mathbf{S_{v_j}}) + D(\mathbf{S_{u[i+1,|S_u|]}}, \mathbf{S_{v[j+1,|S_v|]}})$$

   sia minore dell'estremo di Carrillo-Lipman.

4. Si applica l'algoritmo in programmazione dinamica per allineamenti multipli **solo** per le posizioni (i, j) che soddisfano il punto 3.

Un modo per ovviare al problema della complessità, è quello di far ricorso ad algoritmi di approssimazione, come ad esempio l'algoritmo "Center Star" ([GUS93]) per allineamenti SP-ottimali che possiede un rapporto di approssimazione pari a 2.

5. Matrici dei costi per gli aminoacidi

Tutti gli algoritmi per comparare le sequenze di proteine fanno riferimento ad alcuni schemi che assegnano costi (o pesi) in corrispondenza di ognuna delle 210 possibili coppie di aminoacidi (190 coppie di differenti aminoacidi più 20 coppie di aminoacidi identici). In generale, queste tabelle dei costi sono rappresentate come matrici 20x20 di similarità (dette matrici costo o matrici sostituzione), dove aminoacidi identici e quelli con caratteristiche simili hanno un peso più alto rispetto a quelli con maggiori differenze.
L'assegnazione dei pesi è molto importante nel confronto tra sequenze, infatti questi possono largamente influenzare gli esiti dei nostri confronti. Idealmente, i pesi dovrebbero riflettere i fenomeni biologici che l'allineamento cerca di mostrare.
Introduciamo alcuni schemi di assegnamento pesi:

- *Lo schema dei costi identità*: le coppie di aminoacidi vengono classificate in due specie: *identiche* e *non identiche*.
- *Gli schemi dei costi per similarità fisicochimiche*: vengono assegnati maggiori pesi agli allineamenti di aminoacidi con proprietà fisico-chimiche similari.
- *Gli schemi dei costi per sostituzioni osservate*: i più usati attualmente, sono derivati dall'analisi delle frequenze delle sostituzioni osservate negli allineamenti tra sequenze.

I due schemi per l'assegnamento dei costi per sostituzioni osservate più importanti sono:
-Le matrici nPAM di M. Dayhoff ([DAY]),
-Le matrici BLOSUMn di Steven e Jorja Henikoff ([HH]).

Le differenze principali tra queste due famiglie di matrici sono:

1. Le matrici PAM si basano su un esplicito modello evolutivo (le sostituzioni sono contate sui rami di un albero filogenetico), mentre le matrici BLOSUM sono basate su un modello implicito di evoluzione che non viene espresso formalmente.

2. Le matrici PAM sono basate su mutazioni osservate attraverso allineamenti globali, questo include sia regioni altamente conservate sia regioni altamente mutabili; le matrici BLOSUM invece, sono basate solamente su regioni altamente conservate con allineamenti locali che non consentano gaps.

3. La procedura BLOSUM utilizza gruppi di sequenze nelle quali non tutte le mutazioni vengono contate allo stesso modo: il calcolo tiene conto se la mutazione interviene tra due sequenze appartenenti alla stessa famiglia o no.

Tuttavia questi tipi differenti di matrici possono essere comparati utilizzando una misura di quantità di informazione media per coppie di aminoacidi , in unità bit, detta "entropia relativa. A seguito di tale comparazione , si può concludere che per il confronto tra sequenze strettamente correlate occorre utilizzare le matrici BLOSUM con valori elevati o le matrici PAM con valori bassi; mentre per la comparazione tra sequenze relazionate lontanamente si utilizzano le BLOSUM con valori bassi e le PAM con valori alti.
I moderni applicativi euristici per il confronto tra sequenze (FASTA e BLAST) permettono di scegliere tra varie matrici PAM e BLOSUM in modo tale da permettere ai biologi di scegliere la matrice che possa "pesare" meglio gli allineamenti sottoposti.
In generale comunque, Le matrici utilizzate maggiormente in bioinformatica sono la 250PAM e la BLOSUM62.

Bibliografia

[BU] P. Buneman, *A note on the metric properties of trees*, Journal of Combinatorial Theory (B) Vol. 17 (1974) pagg. 48-50.

[CL] H. Carrillo , D. Lipman , *The multiple sequence alignment problem in biology* , SIAM Journal on Applied Mathematics **48** (1988) pagg 1073-1082.

[DAY] M. O. Dahyoff, *Atlas of protein sequence and structure*, National Biomedical Research Fundation Vol. **11** Suppl.3 (1978) pagg 345-352.

[GUS93] Dan Gusfield, *Efficient methods for multiple sequence alignment with guaranteed error bounds*, Bulletin of Mathematical Biology **55** (1993) pagg. 141 - 154.

[GUS97] Dan Gusfield, *Algorithms on strings, trees and sequences*, Cambridge University Press , New York (1997).

[GJ] M. R. Garey & D. S. Johnson , *Computers and Intractability: a Guide to the Theory of NP-Completeness*, W.H. Freeman & Company (1979) .

[HH] S. Henikoff, J. Henikoff , *Amino acid substitution matrices from protein blocks* , PNAS Classification:Biochemistry **89** (1992) pagg 10915-10919.

[KF] M. K. Kuhner, J. Felsenstein , *A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates* , Molecular Biology and Evolution **11** (1994) pagg 459-468.

[LJW] L. Lawler, T. Jiang L. Wang, *Approximation algorithms for tree alignment with a given phylogeny*, Algorithmica **16** (1996).

[LW] E. S. Lander, M. S. Waterman *Calculating the Secrets of Life*, National Academy Press (1995).

[NW] S. B. Needleman , C. D. Wunsch *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, Journal of Molecular Biology **48** pagg. 443-453 (1970).

[PH] Page R. & Holmes E. *Molecular evolution: a phylogenetic approach*, Blackwell Science, (1998) cap. 2.

[PR] R. Pearson, G. Robins, T. Zhang, *Generalized Neighbor-Joining: More Reliable Phylogenetic Tree Reconstruction*, Journal of Molecular Biology and Evolution **16** (1999) pagg 806-816.

[RB] R. Bellman, *Dynamic programming*, Princeton University Press, (1957).

[SN] N. Saitou, M. Nei, *The Neighbor-Joining method: a new method for recostructing phylogenetic trees*, Molecular Biology and Evolution **4** (1987) pagg. 406-425.

[SS] P. H. A. Sneath , R. R. Sokal *Numerical Taxonomy*, W.H. Freeman & Company (1973) .

[SW] T. F. Smith, M. S. Waterman *Identification of common molecular subsequences*, Journal of Molecular Biology, **147(1)** pagg 195-197 (1981).

[WJ] L. Wang & Jiang , *On the complexity of multiple sequence alignment*, Journal of Computational Biology **1** (1994) pagg 337-338.

[ZP] E. Zuckerkandl, L. Pauling, *Horizons in biochemistry*, Chapter Molecular disease, evolution and genetic heterogeneity M. Marsha, B. Pullman New York pagg. 189-225 (1962).

Siti Web

- [AD] Andreas Dress, *The Mathematical Basis of Molecular Phylogenetics*
  www.techfach.uni-bielefeld.de/bcd/Curric/MathAn/mathan.html
- [KRT] R. Karp, L. Ruzzo, M. Tompa, *Algorithms in molecular biology*
  www.cs.washington.edu/education/courses/590bi/96w/
- NCBI- National Center for biotechnology information
  www.ncbi.nih.gov
- [RS] Ron Shamir, *Algorithms for molecular biology: course archive*
  www.math.tau.ac.il/ ˜rshamir/algmb.html
- [CH] Christian Charras, Thierry Lecroq, *Sequence comparison*
  www.igm.univ-mlv.fr/ ˜lecroc/seqcomp/
- [GF] Georg Fuellen, *A gentle guide to multiple alignment*
  www.csu.edu.au/ci/vol04/mulali/mulali.html