

UNIVERSITA' DEGLI STUDI DI SALERNO



Dipartimento di Informatica

PROGETTO

STRUMENTI FORMALI PER LA BIOINFORMATICA

Clustal: tool per l'allineamento multiplo di sequenze

Anno Accademico 2024-2025

Candidato:

Debora Pucciarelli

Matricola: 0522501933

Professori corso:

Rosalba Zizza

Rocco Zaccagnino

Clelia De Felice

Indice

| | | |
|----------|---|-----------|
| 1 | Allineamento Progressivo | 2 |
| 1.1 | Introduzione | 2 |
| 1.2 | Fasi Principali | 2 |
| 1.3 | Vantaggi e Limitazioni | 3 |
| 1.4 | Conclusione | 3 |
| 2 | Clustal - tool Allineamento progressivo | 4 |
| 2.1 | Introduzione a Clustal | 4 |
| 2.2 | Principali Caratteristiche di Clustal | 4 |
| 2.3 | Vantaggi e applicazioni | 4 |
| 2.4 | Limitazioni | 5 |
| 2.5 | Esecuzione di un assignment su ClustalOmega | 5 |
| 2.5.1 | Ferritina | 5 |
| 2.5.2 | Caso di studio su sequenze di ferritina | 5 |
| 2.5.3 | Passi per l'esecuzione ed analisi dei risultati | 7 |
| 2.5.4 | Rappresentazione allineamento sequenze in forma grafica | 8 |
| 2.5.5 | Analisi risultato Albero Guida | 9 |
| 2.5.6 | Analisi risultato Albero filogenetico | 9 |
| 2.6 | File FASTA | 10 |
| 3 | Evoluzione di Clustal | 11 |
| 3.1 | Introduzione | 11 |
| 3.2 | ClustalV | 11 |
| 3.2.1 | Allineamenti di profili [1] | 11 |
| 3.2.2 | UPGMA | 12 |
| 3.2.3 | Ulteriori caratteristiche | 12 |
| 3.2.4 | Limitazioni | 12 |
| 3.3 | ClustalW | 12 |
| 3.3.1 | Limitazioni | 13 |
| 3.4 | Clustal Omega | 13 |
| 3.4.1 | Alberi guida mBed | 13 |
| 3.4.2 | Modelli Markov nascosti (HMM) | 13 |
| 3.4.3 | Ulteriori caratteristiche | 14 |
| 3.4.4 | Limitazioni | 14 |
| 3.5 | Conclusione - Confronto tra versioni | 14 |
| 4 | Galaxy | 15 |
| 4.1 | Introduzione | 15 |
| 4.2 | Interfaccia Galaxy | 15 |
| 4.2.1 | Caso di studio | 15 |
| 4.2.2 | Analisi Risultato Sequence Logo | 17 |
| 4.3 | Conclusioni | 18 |

| | | |
|----------|--|-----------|
| 5 | Analisi tool ClustalW - Allineamento Multiplo di Sequenze | 19 |
| 5.1 | Introduzione | 19 |
| 5.2 | Flusso esecuzione Allineamento multiplo sequenze | 19 |
| 5.2.1 | Invocazione della funzione profileAlign in multiSeqAlign | 20 |
| 5.2.2 | Applicazione progDiff su profileAlign | 20 |
| 5.2.3 | Funzione progDiff: punteggio allineamento | 21 |
| 5.2.4 | Esempio pratico sul calcolo dei punteggi | 22 |
| 5.3 | Conclusione | 23 |
| | Bibliografia | 24 |

Capitolo 1

Allineamento Progressivo

1.1 Introduzione

L'**allineamento progressivo** è una tecnica utilizzata per eseguire l'allineamento multiplo di sequenze biologiche, come DNA, RNA o proteine, in modo rapido ed efficace.

L'idea di base dell'allineamento progressivo consiste nel costruire un **allineamento multiplo di sequenze (MSA)** in maniera iterativa e gerarchica, partendo dall'allineamento di coppie di sequenze simili e proseguendo gradualmente verso l'allineamento di gruppi sempre più grandi.

In pratica, date le sequenze s_1, s_2, \dots, s_k , queste vengono aggiunte una alla volta a un allineamento multiplo che cresce progressivamente. Si inizia calcolando l'allineamento tra s_1 ed s_2 , poi si aggiunge s_3 , quindi successivamente s_4 , e così via, fino a includere tutte le sequenze. [2]

1.2 Fasi Principali

Per sviluppare il tutto è possibile suddividere il processo in tre fasi principali:

- **Calcolo delle distanze tra sequenze:** viene determinata una *matrice di similarità* tra tutte le coppie di sequenze utilizzando o allineamenti *globali* (Needleman-Wunsch), che considerano l'intera lunghezza delle sequenze, o *locali* (Smith-Waterman), che si concentrano su regioni di alta similarità all'interno delle sequenze. In questa matrice vengono rappresentati i *punteggi di allineamento pairwise* che vengono, successivamente, trasformati in valori che indicano la distanza relativa tra le sequenze;
- **Costruzione di un albero di guida:** *l'albero di guida* è una struttura gerarchica che rappresenta le relazioni tra le sequenze sulla base della matrice di distanza. Tra gli algoritmi principali:
 - **UPGMA:** assume tassi di evoluzione costanti. Ogni sequenza viene vista come un *cluster separato*, che è semplicemente un gruppo che può contenere una o più sequenze. Successivamente, trova e combina i due cluster più vicini (quelli con la distanza più piccola definita dalla matrice delle distanze, esempio A e B) in un nuovo cluster (C). Calcola la distanza tra il nuovo cluster e gli altri nodi (esempio D) usando la *media delle distanze dei due cluster uniti rispetto ad un altro nodo* e si aggiorna la matrice delle distanze. Si ripete il processo fino a che tutti i cluster non sono stati uniti in un unico albero *radicato* identificando un punto di origine definito dell'antenato comune più remoto.
 - **NEIGHBOR-JOINING:** opera nello stesso modo di *UPGMA* ma è più flessibile poichè non assume tassi di evoluzione costanti, ogni sequenza viene vista come un *nodo separato*, cioè un punto isolato che dobbiamo collegare agli altri. Dalla matrice delle distanze, si cerca la coppia di sequenza meno distanti e si uniscono in un nuovo nodo, si aggiorna la matrice delle distanze calcolando la *media delle distanze tenendo conto sia della distanza tra due nodi da unire e sia delle distanze complessive di ciascun nodo rispetto agli altri*. Si ripete il processo fino a quando tutte le sequenze sono collegate in un unico albero di tipo *non radicato* e quindi non ha un punto d'origine.

L'albero guida è strutturato in modo che le sequenze più simili sono vicine nelle foglie, mentre quelle meno correlate sono posizionate più lontano nella struttura in modo da stabilire l'ordine di allineamento progressivo.

- **Allineamento progressivo:** Una volta costruito l'albero guida, il processo di allineamento multiplo inizia *dalle sequenze più simili* identificate come più vicine secondo l'albero in modo da ottenere una base solida e accurata su cui costruire.
Una alla volta vengono aggiunte *progressivamente* le sequenze rimanenti, seguendo l'ordine gerarchico definito dall'albero. Ogni nuova sequenza viene allineata rispetto alle sequenze già allineate.
Successivamente si passa all'*allineamento di gruppi* in cui nelle sequenze già allineate (gruppi) vengono aggiunte progressivamente o altri gruppi o una singola sequenza, seguendo l'ordine gerarchico definito dall'albero, considerando intere colonne come unità e non più le singole lettere.
Infine si calcolano i *punteggi di similarità*, per integrare una nuova sequenza o un nuovo gruppo, considerando tutti i caratteri nelle colonne corrispondenti e le distanze di similarità tra sequenze garantendo coerenza e accuratezza nell'allineamento globale. [3]

1.3 Vantaggi e Limitazioni

L'allineamento progressivo presenta diversi vantaggi, tra cui: [2]

- **Efficienza computazionale:** la sua complessità temporale cresce in modo quadratico rispetto al *numero di sequenze* e alla loro *lunghezza*. Infatti richiede meno risorse rispetto a metodi globali esatti per l'MSA;
- **Riduzione degli errori iniziali**, poiché le sequenze più simili vengono allineate per prime;
- **Individua** efficientemente sequenze particolarmente simili o invarianti.

Per quanto riguarda, invece, le limitazioni:

- Gli **errori** introdotti nelle prime fasi non possono essere corretti successivamente;
- **Non ottimizza** una funzione obiettiva globale, il che può influire sulla qualità dell'allineamento complessivo;
- **Forte dipendenza** con l'accuratezza dell'albero guida.

1.4 Conclusione

Nonostante le sue limitazioni, grazie alla sua efficienza computazionale e alla capacità di trattare dataset di dimensioni moderate, l'*allineamento progressivo* resta un approccio ampiamente utilizzato. Strumenti come **ClustalW**, **MUSCLE** e **T-COFFEE** hanno ulteriormente migliorato la precisione di questo metodo attraverso ottimizzazioni iterative.

In conclusione, l'allineamento progressivo è utile per analizzare sequenze biologiche simili, ma è necessario prestare attenzione nella costruzione dell'albero guida, poiché errori in questa fase possono influenzare significativamente l'allineamento finale.

In seguito, ci concentreremo sulla famiglia di strumenti **Clustal**, in particolare su **Clustal Omega**, per analizzare un caso di studio su sequenze biologiche, focalizzandoci sulle sequenze di *ferritina* e interpretando i risultati ottenuti.

Capitolo 2

Clustal - tool Allineamento progressivo

2.1 Introduzione a Clustal

Dopo aver definito una panoramica di come si effettua l'allineamento progressivo, è fondamentale comprendere come questa tecnica prenda vita nella pratica bioinformatica.

Uno degli strumenti più celebri e affidabili in questo ambito è **Clustal**.

Clustal è un programma progettato per allineare in maniera efficiente e preciso sequenze biologiche, come proteine e DNA. Grazie a una serie di ottimizzazioni, è diventato uno strumento molto usato per chi lavora nel campo della bioinformatica perché riesce, in maniera veloce, ad identificare con molta accuratezza le regioni di somiglianza tra sequenze diverse.

2.2 Principali Caratteristiche di Clustal

Clustal è implementato in modo da effettuare allineamento di sequenze biologiche con **approccio progressivo** particolarmente efficiente, effettuando inizialmente l'allineamento delle sequenze più simili (secondo un albero guida basato sulle distanze pairwise), proseguendo con quelle più distanti, riducendo così la complessità computazionale con risultati di alta qualità.

Tra le caratteristiche distintive che Clustal possiede per effettuare l'allineamento è la capacità di attribuire un **peso** diverso a ciascuna sequenza in modo tale che le sequenze molto simili vengano considerate meno importanti, mentre quelle più diverse acquisiscono maggiore rilevanza, così facendo si ottengono allineamenti affidabili anche quando si analizzano gruppi di sequenze molto diverse.

Inoltre, Clustal è in grado di valutare in modo intelligente dove **inserire i gap** (ossia inserimenti e cancellazioni) nell'allineamento tenendo conto del contesto e della struttura della sequenza, evitando di inserire gap in posizioni che non hanno senso biologico.

Inoltre, il tool utilizza delle **matrici di sostituzione**, come *PAM* e *BLOSUM*, che assegnano un punteggio alla sostituzione di un amminoacido con un altro durante l'evoluzione, rimanendo fisse e invariabili durante il processo di allineamento.[4]

2.3 Vantaggi e applicazioni

Grazie alla sua affidabilità e versatilità, **Clustal identifica** regioni conservate tra sequenze correlate in modo da comprendere la funzione di proteine e acidi nucleici; **individua** la forma tridimensionale di una proteina conoscendo solo la sua sequenza di amminoacidi; e **fornisce** allineamenti di alta qualità per la costruzione di alberi filogenetici che ci permettono di ricostruire l'evoluzione di organismi e molecole.

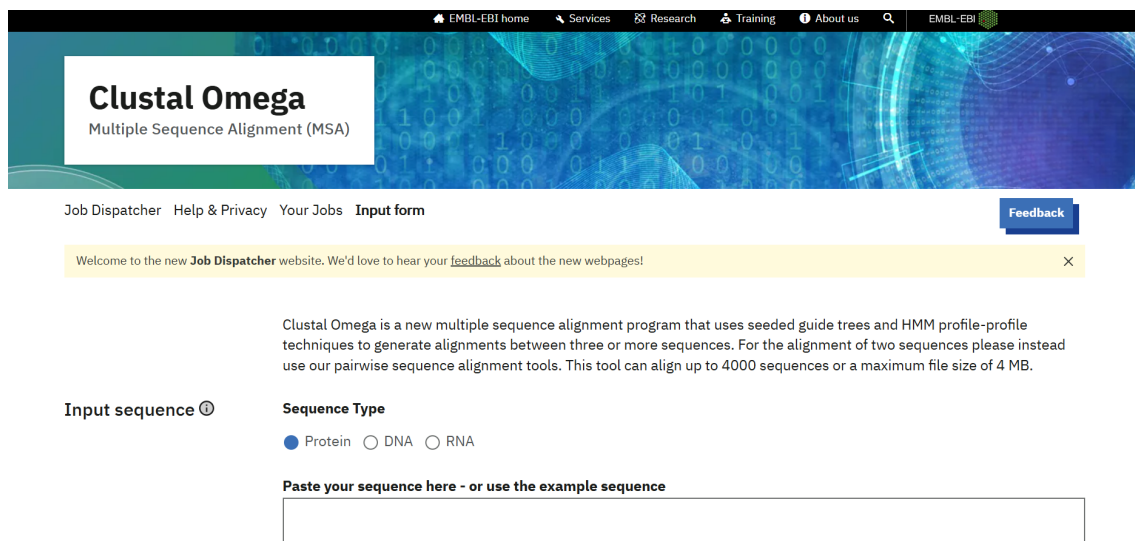
2.4 Limitazioni

Clustal, pur essendo uno strumento molto utilizzato, presenta alcune limitazioni, come *l'impossibilità di correggere errori* introdotti nelle prime fasi e *la dipendenza dall'accuratezza* dell'albero guida.

2.5 Esecuzione di un assignment su ClustalOmega

In questo paragrafo, ci concentreremo su *ClustalOmega*, una versione specifica di Clustal, che verrà analizzata in dettaglio nel capitolo successivo, insieme ad altre varianti.

Per illustrare le sue funzionalità, presenteremo un caso pratico, eseguito dal sito ufficiale di *ClustalOmega* [5]



The screenshot shows the Clustal Omega website interface. At the top, there is a navigation bar with links: EMBL-EBI home, Services, Research, Training, About us, and a search icon. Below the navigation bar, the main header features the Clustal Omega logo and the text "Multiple Sequence Alignment (MSA)". A secondary navigation bar includes links: Job Dispatcher, Help & Privacy, Your Jobs, Input form, and a Feedback button. A yellow banner below the navigation bar reads: "Welcome to the new Job Dispatcher website. We'd love to hear your feedback about the new webpages!". The main content area contains a description of Clustal Omega: "Clustal Omega is a new multiple sequence alignment program that uses seeded guide trees and HMM profile-profile techniques to generate alignments between three or more sequences. For the alignment of two sequences please instead use our pairwise sequence alignment tools. This tool can align up to 4000 sequences or a maximum file size of 4 MB." Below this description, there is a form for inputting sequences. The form has a label "Input sequence ⓘ" and a "Sequence Type" section with radio buttons for Protein (selected), DNA, and RNA. Below the radio buttons, there is a text input field with the placeholder "Paste your sequence here - or use the example sequence".

Figura 2.1: Parte dell'interfaccia grafica del sito Clustal Omega

2.5.1 Ferritina

La *ferritina* è una proteina responsabile del metabolismo del ferro in particolare del *deposito del ferro* e della *protezione cellulare* aiutando a ridurre i danni causati dal *ferro libero*, che può essere pericoloso per le cellule. Il ferro libero non è nient'altro che il ferro presente nella cellula o nel sangue che non è legato a proteine o molecole specifiche.

La ferritina, inoltre, è costituita da 24 *subunità*, ovvero singole unità proteiche assemblate insieme formano la struttura finale della ferritina, all'interno del quale viene immagazzinato il ferro.[6]

2.5.2 Caso di studio su sequenze di ferritina

Consideriamo ora quattro sequenze di ferritina:

- >AAA35832.1 ferritin [*Homo sapiens*]
MTTASTSQVRQNYHQDSEAAINRQINLELYASYVYLSMSYYFDRDDVALKNFAKYFLH-
QSHEEREHAEKL
MKLQNQRGGRIFLQDIKKPDCDDWESGLNAMECALHLEKNVNQSLLELHKLATDKND-
PHLCDFIETHYLN
EQVKAIKELGDHVTNLRKMGAPESGLAEYLFDKHTLGDSDNES
– Sequenza che appartiene alla ferritina di *Homo sapiens* (esseri umani);
- >ORT44433.1 ferritin [*Escherichia coli*]
MATAGMLLKLNSQMNRIFYASNLYLHLSNWCSEQSLNGTATFLRAQAQSNVTQMRRM-
FNFMKSVGATPIV

KAIDVPGEKLSLEELFQKTMEEYEQRSSTLAQLADEAKELNDDSTANFLRDLEKEQ-
 QHDGLLQTLDE
 VRSAGMCPVQTDQHVNLVVSHQLH

- Sequenza che riguarda alla ferritina di *Escherichia coli*, un batterio gram-negativo che si trova nella flora batterica del nostro intestino; [7];
- >NP_803431.1 ferritin, mitochondrial precursor [*Homo sapiens*]
 MLSCFRLLSRHISPSLASLRPVRCCFALPLRWAPGRPLDPRQIAPRRPLAAAASSR-
 DPTGPAAGPSRVQR
 NFHPDSEAAINRQINLELYASYVYLSMAYYFSRDDVALNNFSRYFLHQSREETEHA-
 EKLMLRQNRGGRI
 RLQDIKKPEQDDWESGLHAMECALLLEKNVNQSLLELHALASDKGDPHLCDFLETY-
 YLNEQVKSIELGD
 HVHNLVKMGAPDAGLAEYLFDTHTLGNNKQN
- Sequenza che appartiene alla ferritina *mitocondriale precursore di Homo sapiens*, il cui ruolo è importante nel metabolismo del ferro e nella protezione contro il danno ossidativo all'interno di essi;
- >XJQ89755.1 ferritin [*Clostridioides difficile*]
 MSSYLEAEGLKGFSNWFYVQYKEETDHAMFFYKYLHNVGGKVQLDAIPMPDAGFTS-
 AMDILERTLAHEKK
 VTALINNLAAVANSESDFRTSQFLLWFISEQAEETNCEDNIKRVKLAGEGGLFFVDQE-
 FANRTYAAPT
 PPVTI
- Sequenza che fa parte alla ferritina di *Clostridioides difficile*, un battere gram positivo anaerobio, che produce una tossina, la quale provoca infiammazione dell'intestino e diarrea. [8]

Consideriamo la prima sequenza di ferritina, quella che appartiene all'*Homo Sapiens* e analizziamo la sua struttura:

- **Nome proteina:** *Ferritin*, proteine che consentono il deposito del ferro;
- **Id sequenza:** *AAA35832.1*, definisce il codice univoco di accesso una sequenza registrata in un database proteico, come *NCBI GenBank*;
- **Specie:** *Homo sapiens* descrive che la sequenza appartiene ad un essere umano;
- **Stringhe:** ogni lettera rappresenta un aminoacido.

Lo stesso tipo di analisi può essere fatto anche per le altre sequenze della ferritina sopra elencate.

2.5.3 Passi per l'esecuzione ed analisi dei risultati

Un *primo passo* fondamentale per analizzare l'allineamento delle sequenze consiste nell'utilizzare ClustalOmega, accessibile dal sito ufficiale <https://www.ebi.ac.uk/jdispatcher/msa/clustalo> [5] ed inserire le sequenze da allineare o, caricando il file contenente le varie sequenze oppure, "incollarle" direttamente nella sezione "Input sequence" indicando il tipo di sequenza "Protein".

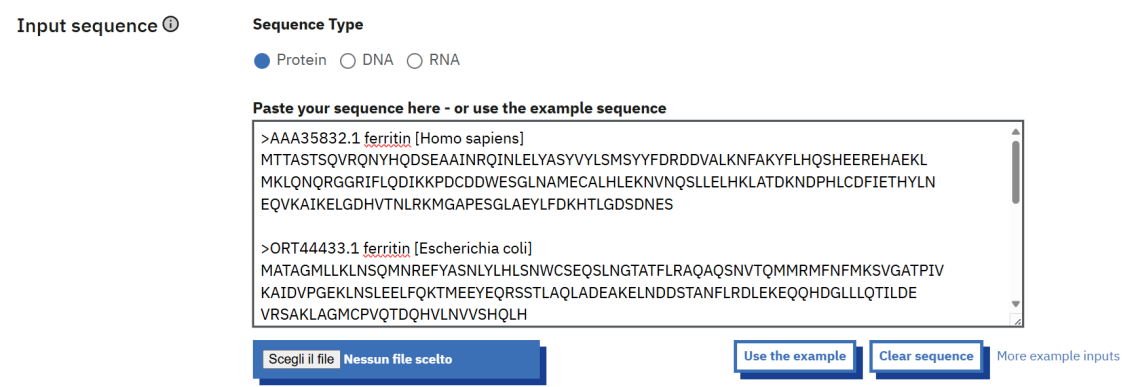


Figura 2.2: Sequenze ferritina dei casi di studio

Per conoscere ed analizzare i risultati dell'allineamento, è necessario sottomettere gli input dal tasto "Submit", attendere e cliccare sul tasto "View Results" per visualizzare i risultati.



Figura 2.3: Risultato Allineamento Sequenze di Ferritina

Analisi risultati

Una prima osservazione che si può fare dopo aver ottenuto i risultati dell'allineamento è l'analisi dei simboli sotto le sequenze che definiscono i **gradi di similarità**. In particolare:

- ***** (asterisco): indica che in quella colonna tutte le sequenze hanno lo stesso amminoacido;
- **:** (due punti): indica che in quella colonna ci sono amminoacidi o nucleotidi diversi, ma chimicamente simili (es. proprietà fisico-chimiche);
- **.** (punto): c'è una similitudine minore rispetto ai due punti (:), infatti potrebbero esserci lettere con proprietà simili, ma non identiche;
- **Nessun simbolo (spazio vuoto)**: non c'è nessuna somiglianza tra i caratteri delle sequenze in quella colonna. [1]

Oltre ai *gradi di somiglianza* tra le sequenze, è possibile notare anche altri aspetti importanti, come i **punteggi**, riportati a destra di ogni riga indicando fino a che posizione della sequenza originale sono mostrati gli amminoacidi; e la presenza di **gaps**, ovvero inserimenti e/o cancellazioni introdotti nell'allineamento in una delle sequenze per *massimizzare* la similarità tra le sequenze, *ottimizzando* l'allineamento *trovando* il maggior numero di corrispondenze e sono rappresentati con trattini (-),

In base a quanto detto, applicato ai risultati ottenuti dal nostro allineamento, possiamo dedurre sia delle zone di *similarità* che *divergenze* tra sequenze.

Similiarità tra sequenze caso di studio

La regione che presenta maggior similarità con un *alto grado* di conservazione, evidenziate dagli asterischi (*), è quella *centrale*, che permette di mantenere la struttura tridimensionale della proteina. Infatti le somiglianze tra le sequenze indicano che probabilmente derivano tutte da una stessa proteina di origine e quindi hanno un antenato comune.

In diverse posizioni, si osservano sostituzioni tra amminoacidi con proprietà chimiche simili, ma nonostante ciò pur cambiando leggermente la sequenza, la funzione globale della proteina rimane invariata.

Divergenze tra sequenze caso di studio

Nelle estremità iniziali delle sequenze *N-terminali*, hanno maggiori cancellazioni e divergenze, suggerendo una scarsa conservazione funzionale di questa regione tra le sequenze.

Nelle estremità finali delle sequenze - *C-terminali*, le sequenze hanno lunghezze e parti finali diverse, il che potrebbe significare che ognuno delle quali svolgono funzioni specifiche.

La regione centrale seppur la più simile tra le sequenze, ci sono alcune differenze nei punti vicini alle parti più importanti. Queste differenze potrebbero influenzare leggermente il modo in cui la proteina funziona.

2.5.4 Rappresentazione allineamento sequenze in forma grafica

Inoltre è possibile visualizzare i risultati gli allineamenti, sia in forma testuale, che grafica, passando nella finestra *"Alignment"* dove zoomando sarà possibile anche visualizzare la sequenza proteica.

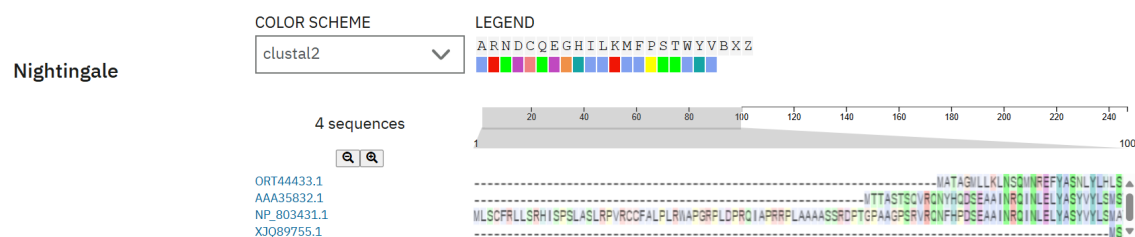


Figura 2.4: Sequenze ferritina dei casi di studio

Le componenti che permettono di definire i risultati di un'analisi di sequenze, identificando somiglianze evolutive, raggruppando sequenze correlate, e effettuando clustering gerarchico, [3] sono l'*albero guida* e l'*albero filogenetico*

2.5.5 Analisi risultato Albero Guida

Per visualizzare l'albero guida, si spostiamo nella sezione *Guide Tree*



Figura 2.5: Rappresentazione risultati albero guida

L'**albero guida** a distanza evolutiva mostra le relazioni tra le sequenze proteiche analizzate, identificate dai codici univoci *ORT44433.1*, *AAA35832.1*, *NP_803431.1* e *XJQ89755.1*, dove le distanze più piccole indicano una maggiore somiglianza evolutiva tra le sequenze. Infatti:

- *AAA35832.1* e *NP_803431.1* sono strettamente correlati, con una distanza evolutiva pari a *0.103825*.
- Il cluster formato da questi due nodi ha una distanza evolutiva di *0.303071* rispetto a *XJQ89755.1*.
- Infine, il nodo radice mostra che *ORT44433.1* è più distante evolutivamente dagli altri nodi.

2.5.6 Analisi risultato Albero filogenetico

Per mostrare, invece, l'albero filogenetico ci spostiamo nella sezione *Phylogenetic Tree*

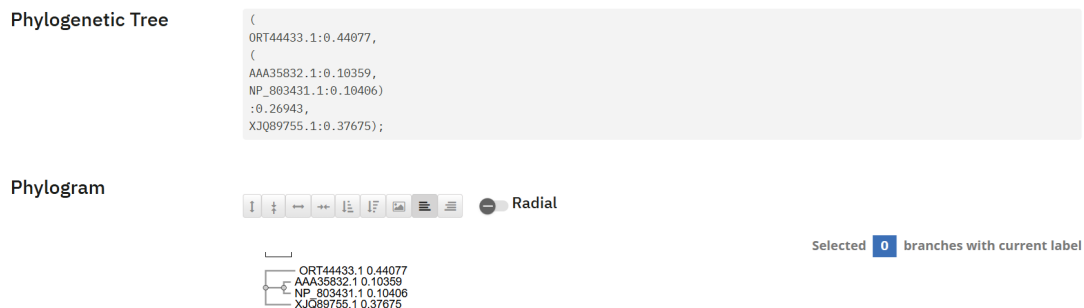


Figura 2.6: Rappresentazione risultato albero filogenetico

L'**albero filogenetico** viene utilizzato, anche esso, per visualizzare relazioni evolutive tra diverse sequenze, considerando sempre i codici univoci che identificano le varie sequenze (*ORT44433.1*, *AAA35832.1*, *NP_803431.1*, *XJQ89755.1*). Ma notiamo delle differenze in particolare:

- *AAA35832.1* e *NP_803431.1* hanno una distanza molto ridotta tra loro di *0.10359* e *0.10406* infatti hanno una stretta correlazione evolutiva;
- Il cluster formato da questi due nodi è distante da *XJQ89755* rispetto a *0.26943*;
- *ORT44433.1*, nodo radice, è più distante evolutivamente dagli altri nodi con una distanza evolutiva di *0.44077* indicando una maggiore divergenza rispetto al resto.

In entrambi gli alberi il **phylogram** permette di rappresentare graficamente le relazioni gerarchiche e le distanze tra i nodi in modo che i rami più corti rappresentano relazioni più strette (come quelle tra *AAA35832.1* e *NP_803431.1*), mentre il ramo più lungo (*ORT44433.1*) indica una separazione evolutiva più marcata.

La **differenza** tra i due alberi (*uno basato su Neighbor-Joining e l'altro su UPGMA*), derivano dall'uso di algoritmi, modelli evolutivi, o parametri di calcolo delle distanze. Queste differenze influenzano il grado di divergenza e i raggruppamenti osservati.

In conclusione, *l'albero filogenetico* mostra una separazione evolutiva più marcata, soprattutto per *XJQ89755.1* e *ORT44433.1*.

Le distanze tra *AAA35832.1* e *NP_803431.1* sono simili in entrambi gli alberi, ma nel primo *XJQ89755.1* è più vicino al cluster.

2.6 File FASTA

Nel menù "*Result Files*" è possibile effettuare il download dell'allineamento in formato *FASTA* delle sequenze proteiche considerate.

I *file FASTA* sono un formato di file comunemente utilizzato per memorizzare ed archiviare grandi quantità di dati biologici in modo efficiente e facilmente leggibile dai programmi di bioinformatica come nel tool *SequenceLogo* di Galaxy, che analizzeremo nel capitolo 4.

Le sequenze ottenute saranno salvate in un file "*ferritina.fa*".

The alignment in FASTA format converted by
Seqret

[clustalo-I20250113-141204-0693-41658457-p1m.fa](#)

Download

Figura 2.7: Download file Fasta

Capitolo 3

Evoluzione di Clustal

3.1 Introduzione

Sebbene *Clustal* è stato uno dei primi strumenti a permetterci di studiare come le proteine si sono evolute nel tempo, le sue prime versioni presentavano alcune limitazioni legate alla potenza di calcolo disponibile. Per far fronte alle crescenti esigenze della ricerca bioinformatica, sono state sviluppate nel tempo nuove versioni, ciascuna delle quali ha introdotto miglioramenti significativi in termini di efficienza, accuratezza e scalabilità. [1]

Di seguito, una panoramica delle principali versioni di Clustal: **ClustalV**, **ClustalW**, e **Clustal Omega**

3.2 ClustalV

ClustalV, introdotto nel 1992, è stato il primo strumento a sfruttare l'approccio progressivo per l'allineamento multiplo di sequenze (**Multiple Sequence Alignment- MSA**) e degli alberi filogenetici introducendo nuove funzionalità che ne hanno ampliato l'utilità e la precisione:

- *Allineamenti di profili*;
- *UPGMA*;

3.2.1 Allineamenti di profili [1]

Prima di dire come funziona un *allineamento tra profili* è necessario introdurre il concetto di **profilo**.

Un *profilo* è una rappresentazione delle frequenze di ciascun carattere, come amminoacidi, in ogni posizione in un allineamento di sequenze, aiutando a capire quali parti delle sequenze sono simili o diverse tra di loro. In altre parole, un profilo raccoglie informazioni su come ogni colonna è composta.

Quando si parla di *allineamento tra profili* si fa riferimento, non più a singole sequenze ma a due o più profili che vengono allineati tra loro confrontando gruppi molto grandi di sequenze in modo più efficiente,

Una volta creati i profili si mettono a confronto in modo da trovare le posizioni dove le lettere si somigliano di più nei gruppi (sequenze già allineate tra loro). Per farlo si utilizzano i *punteggi di similarità* che indicano se le lettere in quella posizione sono molto simili (punteggio alto) o molto diverse (punteggio basso), ottimizzando così l'allineamento.

Una volta che i profili sono allineati, possono essere usati per la costruzione di alberi filogenetici o l'identificazione di regioni conservate o variabili.

ClustalV utilizza la strategia *progressiva* in modo tale che il nuovo allineamento viene costruito gradualmente, seguendo le distanze calcolate nell'albero guida.

Vantaggi allineamenti profili

Questo approccio offre numerosi vantaggi, infatti vi è un notevole *risparmio computazionale*, evitando di ricalcolare tutto ogni volta che si aggiungono nuove sequenze. Inoltre, offre *flessibilità*, permettendo di aggiornare gradualmente gli allineamenti migliorando la *gestione dei dati*. Infine garantisce la *precisione* poichè gli allineamenti esistenti vengono preservati, evitando errori e migliorando la qualità complessiva dei risultati. [1]

3.2.2 UPGMA

In ClustalV, *l'algoritmo UPGMA* viene utilizzato per costruire l'albero guida durante il processo di allineamento delle sequenze ed assume tassi evolutivi costanti ed è basato sulle distanze a coppie tra le sequenze.

Il metodo inizia trattando ogni sequenza come un cluster separato, poi unisce i due cluster più simili in uno nuovo. Calcola la distanza tra questo nuovo cluster e gli altri usando la media delle distanze. Ripete il processo fino a ottenere un unico albero.

3.2.3 Ulteriori caratteristiche

L'*interfaccia* di ClustalV è basata su *riga di comando (CLI)* consentendo di effettuare allineamenti multipli con semplici comandi, senza necessità di intervento manuale. Infatti l'utente interagisce con il programma tramite un terminale o un prompt di comando, dove può specificare vari parametri e file di input.

ClustalV utilizza la matrice di sostituzione **PAM** per calcolare la somiglianza tra le sequenze durante l'allineamento, in particolare viene applicata su sequenze simili, poichè meno adatto per sequenze più distanti in quanto assume che le mutazioni evolutive siano relativamente lente e costanti nel tempo.

3.2.4 Limitazioni

ClustalV, pur essendo una delle prime implementazioni di successo dell'allineamento progressivo di sequenze biologiche, presenta diverse limitazioni quali: [1]

- **Sensibilità agli errori iniziali:** errori iniziali introdotti, come nell'allineamento o nella costruzione dell'albero guida, non possono essere corretti nelle fasi successive, compromettendo la qualità dell'allineamento finale;
- **Metodo basato su una matrice di distanze PAM:** adatta per sequenze evolutivamente simili, ma l'assunzione di evoluzione costante e le limitazioni nel trattare sequenze molto distanti fanno sì che questo metodo non sia sempre efficace;
- **Scalabilità limitata:** ClustalV è poco adatto per dataset di grandi dimensioni, poiché la complessità computazionale aumenta esponenzialmente con il numero di sequenze;
- **Assenza di funzionalità avanzate:** non vi sono funzionalità come penalità di gap specifiche per posizione o scelte automatiche di matrici;

3.3 ClustalW

ClustalW è una versione avanzata di *ClustalV*, introdotta nel 1994, utilizzata per l'allineamento globale di sequenze di acidi nucleici e proteine adottando, anch'esso un metodo progressivo per allineare le sequenze, seguendo l'ordine di ramificazione dell'albero guida[1]. Tuttavia, sono presenti dei miglioramenti a riguardo:

- **Algoritmo Neighbor-Joining:** opera come *UPGMA*, ma durante il calcolo delle distanze tiene conto non solo della distanza tra due nodi, ma anche sulle distanze complessive di ciascun nodo rispetto ad altri. Inoltre non assume un tasso evolutivo costante per tutte le sequenze e questo lo rende più realistico [1];
- **Interfaccia utente:** offre sia un'interfaccia a *linea di comando (CLI)* per gli utenti avanzati, sia *grafica* per un utilizzo più semplice e immediato; [1]

- **Penalità di gap:** sono valori negativi che vengono applicati quando vengono introdotti degli spazi vuoti (gap) nell'allineamento delle sequenze. Si classificano in penalità di *apertura* del gap, che si applica quando si crea un gap in una nuova posizione, e quella per l'*estensione* del gap, che si applica quando un gap già esistente viene allungato. Hanno l'obiettivo di evitare l'inserimento eccessivo di gap, aggiungendoli solo dove necessario per migliorare la corrispondenza tra le sequenze, in particolare in regioni variabili. Inoltre, queste penalità non sono fisse, ma variano in base alla regione di allineamento.
- **Matrici di sostituzione:** a differenza di *ClustalV*, si ha la possibilità di scegliere, in base alle esigenze dell'utente e del tipo di sequenze da allineare, le matrici di sostituzione come *BLOSUM*, utilizzata per allineamenti di sequenze proteiche divergenti, e *PAM* per allineare sequenze più simili.

3.3.1 Limitazioni

Nonostante i numerosi miglioramenti, *ClustalW* ha alcune limitazioni, in particolare gli errori nelle prime fasi non possono essere corretti in seguito come nel caso di *ClustalV*, ed inoltre la velocità di esecuzione del tool diminuisce significativamente se si lavora con dataset di dimensioni enormi. Nonostante ciò, è stato un punto di riferimento per molti anni, fino all'introduzione del tool ***Clustal Omega***, che è in grado di gestire sequenze molto lunghe e diversificate.

3.4 Clustal Omega

Clustal Omega, lanciato nel 2011, rappresenta l'ultima versione degli strumenti Clustal progettato per analizzare dataset di grandi dimensione e più complessi, fornendo risultati di migliore qualità in tempi più rapidi [9], e per allineare sequenze di lunghezza variabile con gap interni moderati.[10] Tra le funzionalità principali aggiunte in ***Clustal Omega*** che migliorano la sensibilità, la qualità dell'allineamento e la costruzione dell'albero guida senza compromettere l'accuratezza [10] sono l'utilizzo di:

- *Albero di guida mBed*
- *Modelli Markov nascosti (HMM)*

3.4.1 Alberi guida mBed

Gli ***alberi guida mBed***, seppur funzionano in modo simile agli alberi UPGMA e NJ, vi è una differenza nel calcolo delle distanze, infatti non vengono calcolate tutte le distanze a coppie di sequenze ma si scelgono quelle più rappresentative, ovvero quelle che danno una visione globale del dataset, tutte diverse tra loro, e si calcola la distanza tra queste, velocizzano così la complessità della costruzione a $O(N \cdot \log N)$, dove N indica il numero di sequenze.

Gli alberi guida determinano l'ordine degli allineamenti progressivi, ordinando le sequenze in una struttura gerarchica.[11]

3.4.2 Modelli Markov nascosti (HMM)

I ***Modelli Markov Nascosti- HMM*** sono modelli che migliorano la qualità degli allineamenti di sequenze, specialmente in sequenze complesse e lunghe, considerano non solo gli amminoacidi visibili, ma anche "*stati*" nascosti ovvero condizioni interne del sistema, che potrebbero influenzare l'allineamento.

Il processo inizia identificando le posizioni che probabilmente appartengono alla stessa struttura e allinea le sequenze basandosi su queste informazioni nascoste. Anche se ci sono variazioni nei singoli amminoacidi tra le sequenze che non alterano in modo significativo la struttura complessiva della proteina, vengono accettati rendendo l'allineamento più flessibile.

Per far ciò gli *HMM* considerano sia della probabilità che una sequenza di amminoacidi passi da uno stato nascosto a un altro (*transizioni*) e sia la probabilità di osservare un certo amminoacido in un determinato stato (*emissione*). [11]

3.4.3 Ulteriori caratteristiche

- Clustal Omega è ottimizzato per *sistemi multi-core*, eseguendo operazioni parallele su più core di un processore in modo da distribuire i calcoli necessari per l'allineamento su più unità di elaborazione, rendendolo più veloce l'esecuzione quando si lavora con grandi quantità di dati o sequenze lunghe rispetto a ClustalW, che non supporta il multithreading.
- *Clustal Omega* mette a disposizione sia un'**interfaccia a linea di comando (CLI)** consentendo agli utenti di eseguire allineamenti direttamente nel terminale e sia tramite piattaforme come l'*EBI (European Bioinformatics Institute)*, dove gli utenti possono caricare le loro sequenze e ottenere risultati direttamente attraverso un'interfaccia web. [10]

3.4.4 Limitazioni

Nonostante i suoi vantaggi e la gestione di dataset di grandi dimensione, nel caso in cui essi sono mal allineati o molto grandi, il tool può non completare il calcolo o presentare errori [10], inoltre non è ottimale per sequenze con grandi inserzioni o delezioni interne poiché le penalità sui gap e l'approccio progressivo non sono abbastanza flessibili per gestire queste variazioni strutturali, portando a allineamenti imprecisi.

3.5 Conclusione - Confronto tra versioni

| Caratteristica | ClustalV (1992) | ClustalW (1994) | Clustal Omega (2011) |
|----------------------------|---|---|--|
| Scopo principale | Primo strumento di allineamento progressivo di sequenze | Miglioramento della precisione e flessibilità, introduzione di penalità dinamiche per gap e interfaccia grafica | Scalabilità estrema, progettato per grandi dataset e allineamenti avanzati basati su HMM |
| Metodo di clustering | UPGMA | NEIGHBORD-JOINING (NJ) | APPROCCIO PROBABILISTICO |
| Penalità per gap | - | Dinamiche: variano in base al contesto e alla posizione | Dinamiche avanzate rispetto a ClustalW |
| Matrici di sostituzione | PAM | PAM e BLOSUM | HMM + PAM/BLOSUM |
| Capacità di gestione | Adatto a dataset di piccola dimensione | Adatto per dataset di dimensioni intermedie | Adatto per dataset di grandi dimensioni |
| Interfaccia utente | Solo riga di comando | Riga di comando + Interfaccia grafica | Riga di comando + Interfaccia grafica (EBI) |
| Prestazioni computazionali | Complessità elevata, poco efficiente | Migliorata rispetto a ClustalV | Ottimizzata nel ridurre i tempi di elaborazione |
| Limitazioni | Sensibile agli errori iniziali, non scalabile | Sensibile agli errori iniziali, scalabilità limitata | Per dataset mal allineati può presentare errori |

Figura 3.1: Confronto tra le varie versioni di Clustal

In conclusione:

ClustalV: ha introdotto l'approccio progressivo di sequenze, ma con limitazioni in termini di precisione e scalabilità adottando dataset semplici e piccoli.

ClustalW: ha migliorato significativamente ClustalV, introducendo metodi come NJ per la generazione di alberi filogenetici, le penalità dinamiche per i gap, e un'interfaccia grafica rendendolo più accessibile e flessibile.

Clustal Omega: rappresenta un passo in avanti per la bioinformatica ottenendo una maggior scalabilità, inoltre gestisce dataset di dimensione elevata e utilizza di algoritmi avanzati come il modello HMM.

Capitolo 4

Galaxy

4.1 Introduzione

Galaxy è una piattaforma open-source di bioinformatica che permette di eseguire analisi di dati biologici attraverso un'interfaccia grafica semplice e intuitiva, senza dover scrivere codice. È pensata per essere utilizzata da ricercatori, biologi e bioinformatici, indipendentemente dal livello di esperienza, e consente di analizzare facilmente sequenze genomiche, proteiche.

4.2 Interfaccia Galaxy

Per accedere all'interfaccia principale del tool *Galaxy*, disponibile online, è necessario visitare il sito ufficiale all'indirizzo <https://usegalaxy.org/> [12].

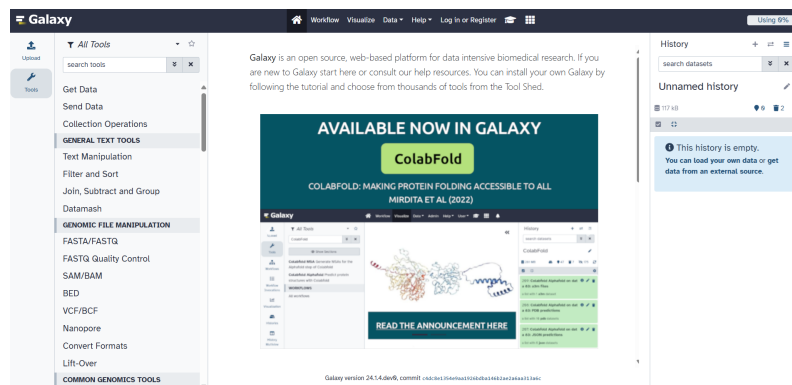


Figura 4.1: Interfaccia tool Galaxy

Galaxy, si può vedere come un "contenitore" di tool selezionabili dall'utente, in base all'attività che intende seguire. Essi sono accessibili dal pannello a sinistra dell'interfaccia e possono essere applicati a specifici set di dati che possono essere caricati direttamente, selezionati dal pannello centrale o scelti tra i risultati delle analisi precedenti visualizzati nella *Cronologia*. [12]

4.2.1 Caso di studio

Riprendendo il caso di studio riguardante le quattro sequenze della ferritina, precedentemente analizzate nel *Capitolo 2* utilizzando il tool *ClustalOmega*, il file in formato FASTA ottenuto verrà utilizzato come input nella piattaforma *Galaxy*.

Come primo passo si cerca "*SequenceLogo*" nei tool di *Galaxy* e si effettua l'*upload* del file FASTA scaricato in precedenza (*ferritina.fa*).

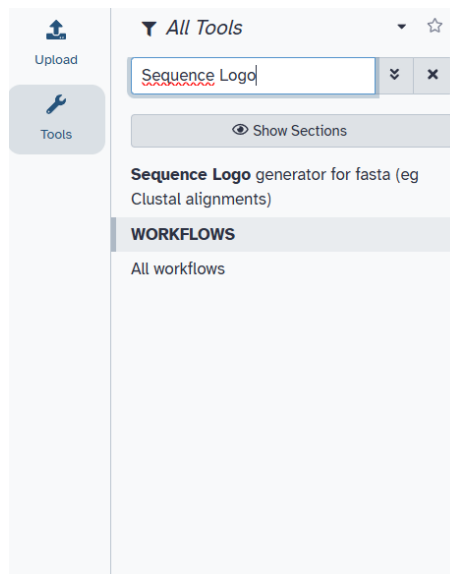


Figura 4.2: Ricerca tool Sequence Logo in Galaxy

Per verificare che il caricamento sia andato a buon fine, è necessario controllare che nella sezione *"History"* sia contenuto il file.

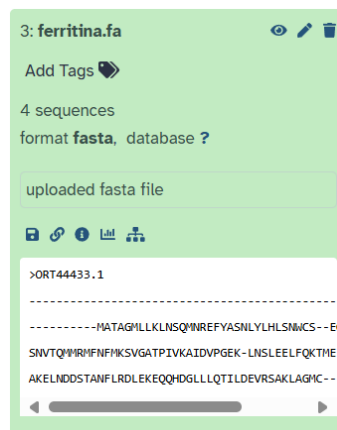


Figura 4.3: Upload corretto file ferritina.fa

Successivamente, cliccare sul pulsante *"Run Tool"* nella sezione centrale dell'interfaccia. Il risultato verrà visualizzato nella sezione destra e potrà essere scaricato in formato *PNG*.

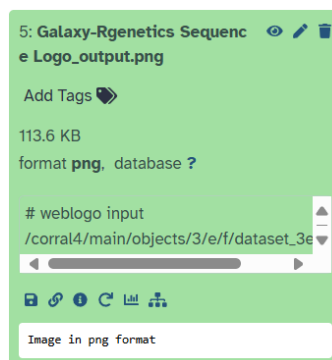


Figura 4.4: Output - run tool

Una volta scaricata l'immagine otterremo il seguente risultato:

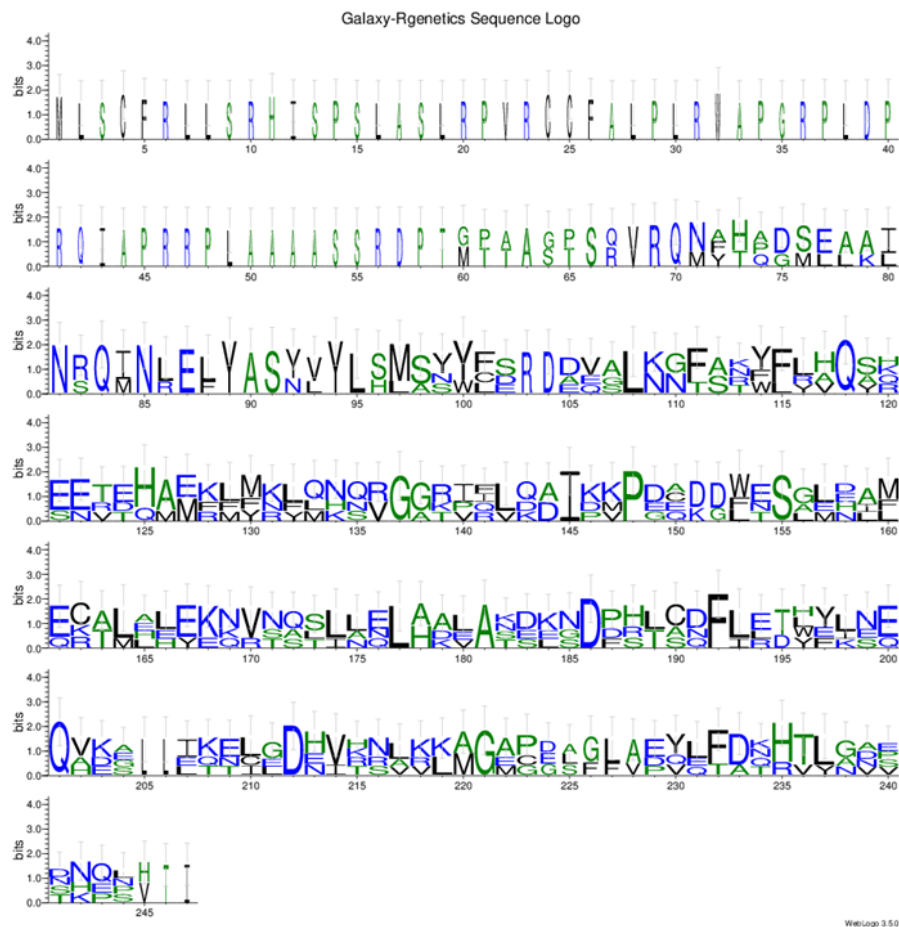


Figura 4.5: Output finale

4.2.2 Analisi Risultato Sequence Logo

L'output ottenuto è un **Sequence Logo**, una rappresentazione grafica generata tramite Galaxy per visualizzare l'allineamento di sequenze proteiche di ferritina. Questo tool permette di mostrare in modo chiaro e conciso le posizioni conservate e variabili all'interno di un allineamento multiplo di sequenze biologiche.

La *struttura generale* del Sequence Logo evidenzia la **conservazione** e la **variabilità** degli amminoacidi all'interno di una sequenza allineata.

L'altezza di ciascuna lettera, che rappresenta un amminoacido, è proporzionale alla sua frequenza in una posizione specifica: più alta è la lettera, più frequentemente quell'amminoacido appare in quella posizione, indicando una forte **conservazione** evolutiva definita da **motivi conservati**.

Al contrario, lettere più piccole segnalano posizioni con maggiore **variabilità**, suggerendo che la variazione in quel particolare amminoacido non ha un grande impatto sul funzionamento della proteina o sull'adattamento dell'organismo, quindi non influisce negativamente verso la sua conservazione.

Dall'output ottenuto possiamo evincere:

- una **moderata conservazione** nella regione *iniziale* dei residui amminoacidici;
- un'**alta conservazione** dei residui di amminoacidici nella regione *centrale*;
- un'**alta variabilità** dei residui di amminoacidici nelle *ultime* posizioni (regione finale);

4.3 Conclusioni

In questo capitolo si è mostrato l'uso della piattaforma *Galaxy* per analizzare sequenze proteiche, concentrandoci sul caso di studio della ferritina. Dopo aver caricato il file di sequenza, abbiamo utilizzato lo strumento *Sequence Logo* di Galaxy per visualizzare un allineamento multiplo delle sequenze. L'output ottenuto, ha permesso di identificare in modo chiaro le aree di alta conservazione e variabilità degli amminoacidi nelle diverse posizioni.

Capitolo 5

Analisi tool ClustalW - Allineamento Multiplo di Sequenze

5.1 Introduzione

Questo capitolo descrive in dettaglio del flusso di esecuzione e le funzionalità principali del software ClustalW per l'allineamento multiplo di sequenze (MSA).

Per condurre ciò si effettua il download della versione più recente di ClustalW dal sito ufficiale <http://clustal.org/clustal2/#Download>. [13]

5.2 Flusso esecuzione Allineamento multiplo sequenze

L'allineamento multiplo tra sequenze, viene eseguito tramite la funzione *multiSeqAlign* in MSA.cpp utilizzando una matrice di distanze per stabilire somiglianze tra le sequenze.

Il processo segue diverse fasi che basate su funzioni interconnesse:

- *multiSeqAlign*: è una funzione principale che gestisce l'allineamento progressivo organizzando le sequenze secondo una matrice di distanze e avviando il processo iterativo di allineamento;
- *profileAlign*: effettua l'allineamento tra gruppi di sequenze rappresentati come profili;
- *progDiff*: ha lo scopo di calcolare il miglior allineamento tra due sequenze o profili utilizzando un approccio divide-et-impera.

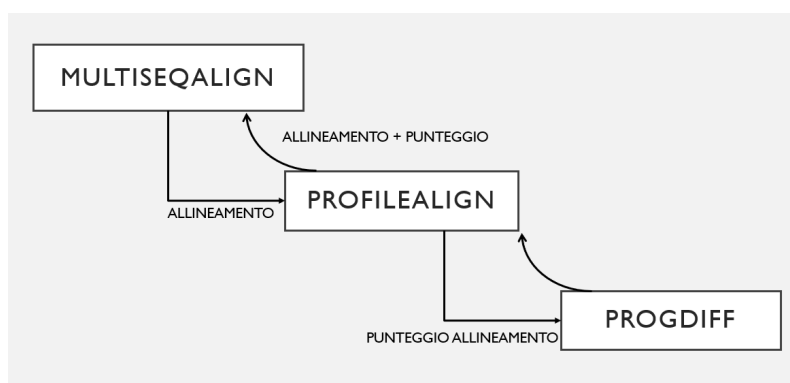


Figura 5.1: Flusso esecuzione allineamento multiplo tra sequenze

5.2.1 Invocazione della funzione `profileAlign` in `multiSeqAlign`

L'inizio del processo dell'allineamento multiplo è dato dalla funzione `multiSeqAlign` che prende in input un insieme di sequenze ed utilizza una matrice delle distanze per identificare quali sequenze sono più simili tra loro e le raggruppa in base alla loro divergenza.

Come noto, l'allineamento multiplo inizia il suo processo con le sequenze più simili usando un algoritmo di programmazione dinamica, invocando `profileAlign`, formando un primo **gruppo allineato**.

Successivamente, `multiSeqAlign` seleziona la sequenza successiva dalla matrice delle distanze e richiama nuovamente `profileAlign`, per allineare la nuova sequenza con il profilo esistente.

```
score = alignAlgorithm->profileAlign(alnPtr, distMat, progSteps->getStep(set), aligned);
```

Figura 5.2: Chiamata della funzione `profileAlign` su `multiSeqAlign`

Il puntatore `alignAlgorithm` viene utilizzato per chiamare il metodo **`profileAlign`** che accetta i seguenti parametri:

- *alnPtr*: oggetto che rappresenta un profilo tenendo traccia del gruppo di sequenze già allineate su cui eseguire l'allineamento;
- *distMat*: indica la matrice distanze utilizzata per determinare l'ordine di allineamento;
- *progSteps->getStep(set)*: indica quali sequenze devono essere allineate in quel determinato passaggio dell'allineamento progressivo;
- *aligned*: array che tiene traccia dello stato delle sequenze già allineate.

Dopo l'allineamento, `profileAlign` aggiorna il puntatore `alnPtr` con il nuovo allineamento che sarà immediatamente visibile anche in `multiSeqAlign`. Dunque l'unico valore restituito è il punteggio dell'allineamento.

A questo punto, `multiSeqAlign` identifica la successiva sequenza da allineare utilizzando la matrice delle distanze e ripete il processo, per tutte le sequenze restanti.

Alla fine di tutte le iterazioni, `multiSeqAlign` riceve i punteggi parziali da `profileAlign` che vengono sommati per ottenere il punteggio totale il quale rappresenta la qualità complessiva del risultato: più è alto allora si ha una maggiore similarità tra le sequenze, mentre un punteggio basso segnala discrepanze.

5.2.2 Applicazione `progDiff` su `profileAlign`

La funzione `profileAlign`, definita nel codice sorgente `MyersMillerProfileAlign.cpp`, una volta ricevuti in input i componenti da allineare, avvia la preparazione per l'allineamento: costruisce una *matrice di allineamento/ di profilo* in cui ogni riga rappresenta una sequenza, mentre ogni colonna indica una posizione nell'allineamento, se le sequenze hanno lunghezze diverse, si aggiungono dei gap; determina le *lunghezze dei profili*, e se il profilo contiene più sequenze, calcola la lunghezza media per normalizzare l'allineamento; seleziona la *matrice di sostituzione* più adatta; recupera i parametri di *penalità per i gap* e prepara gli array per il calcolo dei punteggi.

Fatto ciò, `profileAlign` non esegue immediatamente l'allineamento, perchè è necessario determinare il miglior modo per allineare la nuova sequenza. Per questo motivo viene invocata la funzione `progDiff` che determina il miglior punto di allineamento (midpoint) e ne calcola il punteggio ottimale.

La funzione `progDiff` prende in input gli *indici* di inizio della sequenza e del profilo; la *lunghezza effettiva* delle due parti da allineare e i valori iniziali e finali per il calcolo delle *penalità di gap del profilo*.

```
score = progDiff(sb1, sb2, se1 - sb1, se2 - sb2, (*profile1)[0][GAPCOL], (*profile1)[prfLength1][GAPCOL]);
```

Figura 5.3: Chiamata della funzione `progDiff` in `profileAlign`

Una volta ottenuto il punteggio, viene utilizzato da *profileAlign* per ricostruire l'allineamento vero e proprio attraverso la funzione *progTracepath*, combinando le informazioni ottenute da *progDiff* partendo dal *midpoint* e risalendo all'indietro, per determinare il miglior allineamento possibile.

In questa fase si confrontano le varie posizioni dei componenti da allineare: se *coincidono* le allinea; se invece c'è una mancata corrispondenza o una corrispondenza parziale, si valuta se *inserire gap*, che avviene solo se migliora il punteggio complessivo o se la penalità associata è inferiore rispetto a quella derivante da mismatch.

In alcuni casi, una lettera può essere *spostata* in avanti o indietro per ottimizzare l'allineamento e questo accade sia nel caso in cui il costo di un mismatch è più elevato rispetto allo spostamento e sia nel caso in cui lo spostamento di una lettera evita l'inserimento di più gap.

Inoltre, il risultato dell'allineamento viene incorporato nel profilo esistente, che sarà poi passato come input alle iterazioni successive.

Infine viene restituito il punteggio a *multiSeqAlign*.

5.2.3 Funzione progDiff: punteggio allineamento

Lo scopo principale della funzione *progDiff*, anche essa definita nella classe *MyersMillerProfileAlign*, è l'individuazione del miglior *midpoint* dell'allineamento, che rappresenta la posizione che massimizza la somiglianza tra due sequenze o profili, suddividendo il problema in due metà gestibili ricorsivamente.

Determinazione del miglior punto intermedio (midpoint) nella matrice di allineamento

Inizialmente vengono trattati i casi base in cui il problema è ridotto al minimo possibile: se una delle sequenze è vuota l'altra viene allineata completamente con gap ed il punteggio dell'allineamento viene calcolato considerando solo le penalità per i gap.

Se entrambe le sequenze contengono lettere, la sequenza più lunga viene divisa in due parti (*midi*) per semplificare il problema in sotto-problemi più piccoli e supponiamo che la prima sequenza venga divisa.

Durante questa suddivisione, vengono calcolati i punteggi in due fasi: *in avanti* e *all'indietro*.

- **Fase in avanti:** *HH*: tiene traccia dei massimi punteggi accumulati per allineare la prima metà della prima sequenza con la seconda sequenza considerando match, mismatch e penalità, calcolandoli progressivamente dall'inizio delle sequenze verso il centro.
DD: memorizza i punteggi migliori considerando solo i gap aperti nella prima metà della sequenza;
- **Fase all'indietro:** *RR* opera come *HH* ma calcola i punteggi iniziando dalla fine e procedendo verso il centro considerando match, mismatch e penalità, mentre *SS* opera come *DD* considerando solo penalità gap.

Si combinano i punteggi calcolati in modo da determinare il miglior punto di divisione **midpoint** dell'allineamento, ossia il punto che massimizza il punteggio di allineamento totale fino a quel momento.

Analisi parte codice sorgente midpoint

La variabile *midh* tiene traccia del miglior punteggio accumulato finora, inizialmente impostato alla somma dei punteggi nella prima colonna. La variabile *midj*, inizialmente impostata a 0, rappresenta l'indice della colonna associata al miglior punteggio.

Si esegue un ciclo che itera per tutta la lunghezza della seconda sequenza. Ad ogni iterazione, i punteggi *HH* e *RR* vengono sommati per ogni colonna. Se la somma è maggiore o uguale al valore attuale di *midh* allora viene valutata anche la seconda condizione: se $hh[j] > midh$, oppure se $HH[j] != DD[j]$ e $RR[j] == SS[j]$ allora *midh* e *midj* vengono aggiornati per riflettere il miglior punteggio finora trovato.

La condizione aggiuntiva gestisce i casi in cui più colonne hanno lo stesso punteggio massimo, garantendo che nella fase in avanti non sia stato usato un gap ($HH[j] != DD[j]$), mentre nella fase all'indietro sia già presente un gap ($RR[j] == SS[j]$).

Questo permette di evitare midpoint che introdurrebbero gap inutili, migliorando la qualità dell'allineamento.


```

midh = HH[0] + RR[0];
midj = 0;
for (j = 0; j <= N; j++)
{
    hh = HH[j] + RR[j];
    if (hh >= midh)
    if (hh > midh || (HH[j] != DD[j] && RR[j] == SS[j]))
    {
        midh = hh;
        midj = j;
    }
}

```

Figura 5.4: Determinazione midpoint

In alcuni casi, per migliorare la scelta del *midpoint*, *midj* viene scelto in base alle sole penalità dei gap in particolare vengono sommati gli array *DD[j]*, *SS[j]* e *gS[j]* dove *gS[j]* indica il costo di introdurre un gap nella posizione *j* della prima sequenza.

Una volta individuato il midpoint, la funzione *progDiff* richiama sé stessa in modo ricorsivo su ciascuna metà delle sequenze ovvero la prima metà della prima sequenza con la parte ottimale della seconda sequenza (fino a *midj*) e la seconda metà della prima sequenza con il resto della seconda sequenza e durante ogni ricorsione, vengono calcolati i punteggi parziali identificando i rispettivi midpoint.

Al termine della ricorsione vengono combinati i vari punteggi parziali per ottenere il punteggio ottimale dell'allineamento.

5.2.4 Esempio pratico sul calcolo dei punteggi

Consideriamo un gruppo allineato di sequenze di questo tipo:

| | | | | |
|---|---|---|---|---|
| | A | T | C | G |
| A | | T | - | G |
| A | | T | T | G |

La matrice di sostituzione assegna i punteggi ai *match*: $+1$, ai *mismatch*: -1 e alle penalità per i gap (per *apertura* -2 , per *estensione* -1).

Calcoliamo ora per ogni colonna (analogia *progDiff* sotto-sequenze da confrontare) i punteggi confrontando coppie di righe (singola sequenza) come:

- Nella prima posizione tutti i caratteri corrispondono (*match*) e il punteggio sarà pari a $1+1+1=3$;
(ogni confronto rappresenta la fase in avanti e indietro che calcolano i punteggi per ogni colonna separatamente e si combinano)
- Lo stesso vale per la seconda posizione, quindi il punteggio anche in questo caso è 3 ;
- Nella terza colonna confrontiamo ogni carattere della colonna:
 - C non coincide con T = -1 *mismatch*;
 - C non coincide con - = -2 *apertura gap*;
 - - non coincide con T = -2 *apertura gap*;

Il punteggio parziale sarà $-1-2-2 = -5$;

- Nell'ultima posizione si opera come nella prima o seconda e quindi il punteggio che otteniamo sarà pari a 3

Sommando tutti i punteggi parziali ottenuti avremo come punteggio totale: $3+3-5+3 = 4$ il che indica una discreta similarità tra le sequenze.

5.3 Conclusione

In conclusione, l'allineamento multiplo di sequenze con ClustalW è un processo articolato che si sviluppa attraverso diverse fasi collegate tra loro.

Il primo passo è l'avvio dell'allineamento progressivo tramite la funzione *multiSeqAlign*, che usa una matrice delle distanze per raggruppare le sequenze simili.

La funzione *profileAlign* si occupa dell'allineamento vero e proprio tra le sequenze o i profili, mentre *progDiff* trova il punto di allineamento ottimale, migliorando la qualità complessiva.

Il calcolo dei punteggi, che tiene conto del midpoint e delle penalità per i gap, permette di ottenere un'indicazione chiara della similarità tra le sequenze.

Questo flusso di lavoro consente di ottenere allineamenti accurati e di alta qualità, essenziali per studi filogenetici e comparazioni tra sequenze biologiche.

Bibliografia

- [1] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T. J. Gibson, D. G. Higgins, and J. D. Thompson, "Multiple sequence alignment with the clustal series of programs," *Nucleic Acids Research*, vol. 31, no. 13, 2003.
- [2] "Sequence analysis." https://gi.cebitec.uni-bielefeld.de/_media/teaching/2020summer/sa/sequence_analysis.pdf.
- [3] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Research*, vol. 22, no. 22, 1994.
- [4] "Clustal." <https://www.sciencedirect.com/science/article/abs/pii/S0378111988903307>.
- [5] "Clustal omega." <https://www.ebi.ac.uk/jdispatcher/msa/clustalo>,.
- [6] "Clostridioides difficile." <https://www.my-personaltrainer.it/salute/ferritina.html>,.
- [7] "Escherichia coli." <https://www.hsr.it/news/2022/novembre/escherichia-coli-cos-e-sintomi-cura>,.
- [8] "Clostridioides difficile." <https://www.nurse24.it/studenti/patologia/infezione-da-clostridium-difficile.html>,.
- [9] "Clustalomega: innovazioni e caratteristiche principali." <https://www.ebi.ac.uk/jdispatcher/docs/faqs/clustal/?h=clustal+omega>,.
- [10] "Clustalomega." <https://help.geneious.com/hc/en-us/articles/360044627712-Which-multiple-alignment-algorithm-should-I-use>,.
- [11] "Clustalomega: documentazione ufficiale." <https://www.ebi.ac.uk/>,.
- [12] "Galaxy: documentazione ufficiale." <https://usegalaxy.org/>,.
- [13] "Codice sorgente clustalw v-2.1." <http://clustal.org/clustal2/#Download>,.