UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

APPLICAZIONI DELL'ALGORITMICA ALLA BIOLOGIA: ALBERI EVOLUTIVI

APPLICATIONS OF ALGORITHMICS TO BIOLOGY: EVOLUTIONARY TREES

MATTEO TORTOLI

MARIA CECILIA VERRI

Anno Accademico 2018-2019

CONTENTS

LIST OF FIGURES

INTRODUZIONE

Con la presente tesi si intende fornire una panoramica sulla bioinformatica, una delle scienze emergenti degli ultimi anni. L'obiettivo è quello di mostrare le applicazioni degli algoritmi alla biologia, in particolar modo lo studio e la costruzione degli *alberi evolutivi*, che sono l'argomento principale.
Quest'ultimi, definiti anche *alberi filogenetici*, sono una risorsa chiave sia nell'informatica che nella biologia in quanto permettono lo studio delle relazioni tra le entità biologiche e mostrano la loro evoluzione nel tempo.
L'idea di base della struttura dei capitoli è quella di mostrare, innanzitutto, i concetti base di biologia, necessari per comprendere la materia in oggetto, dopodiché viene presentata la bioinformatica e le sue aree di ricerca, infine vengono illustrati gli algoritmi più importanti per la costruzione degli alberi evolutivi. Di seguito si mostra una panoramica di tali capitoli.

1. *Capitolo 1*: nozioni biologiche di base. Viene dato uno sguardo generale alla classificazione delle forme di vita, dai polisaccaridi fino agli acidi nucleici. Le sezioni successive mostrano il DNA, RNA e proteine. Tutti i concetti di questo capitolo sono funzionali alla comprensione della suddetta tesi. Senza di questi, infatti, non sarebbe possibile capire né contesto né senso di questo studio.

2. *Capitolo 2*: introduzione alla bioinformatica. Le sezioni successive, oltre a dare una definizione della materia e a raccontare la sua storia, dalla sua nascita fino ai giorni nostri, illustrano le principali aree di ricerca, dandone una breve spiegazione a ciascuna di essa. Gli alberi evolutivi fanno parte della cosiddetta filogenetica, in quanto studia le relazioni evolutive tra le entità biologiche attraverso la costruzione di tali alberi.

3. *Capitolo 3*: viene mostrato l'albero evolutivo, cosa è, a cosa serve e che tipi di alberi si possono costruire. Si parla di albero radicato ed albero non radicato, entrambi con funzioni diverse, in quanto il primo mostra le evoluzioni delle entità biologiche, mentre il secondo mostra come sono legate tra loro. Tale capitolo svolge

un ruolo chiave, in quanto vengono poste le basi per il capitolo successivo, che è il "cuore" della presenti tesi.

4. *Capitolo 4*: all'inizio di tale capitolo viene posto il problema che sta alla base degli alberi, ovvero il cosiddetto *problema della costruzione degli alberi evolutivi*. Le sezioni successive mostrano come è possibile risolverlo: di volta in volta vengono mostrati algoritmi sempre più efficaci. Quindi si parte dall'idea di base, poi vengono evidenziate le criticità ed infine vengono sviluppati nuovi algoritmi volti a risolvere tali criticità. Il risultato finale di questa ricerca sono due algoritmi, il *Neighbor-Joining* ed il *Unweighted Pair Group Method with Arithmetic Mean (UPGMA)*, dove il primo permette la costruzione di alberi senza radice, mentre il secondo con radice.
   Tutti gli algoritmi prendono in input una matrice D definita *matrice delle distanze*, in quanto ottenuta calcolando la distanza tra le coppie di elementi di cui è composta.

Esempio di applicazione

Perché gli alberi evolutivi sono così importanti? Perché grazie ad essi è possibile conoscere l'evoluzione delle specie e le relazioni tra le entità biologiche. Per entità non si intendono solamente gli animali o le piante, ma anche la classificazione di virus.
Nei primi anni duemila in Oriente, in particolar modo ad Hong Kong, in Cina, si è espansa a macchia d'olio una malattia infettiva, che nei casi più gravi si è rivelata fatale, ovvero la *SARS* (Severe acute respiratory syndrome) che sta per "sindrome acuta respiratoria grave".
Gli scienziati hanno scoperto che questa malattia è causata da un virus chiamato Coronavirus, che attacca le vie aree sia degli animali che degli umani. All'inizio si era ipotizzato che il virus fosse stato trasmesso agli umani da parte di alcune specie di animali. Ma quali sono questi animali e come è stato possibile?
Attraverso la costruzione dell'albero evolutivo del Coronavirus è stato individuato il responsabile della trasmissione. Nello specifico sono state confrontate le sequenze del virus presente negli animali con quelle degli umani. La corrispondenza migliore è stata ottenuta con la civetta delle palme, un piccolo mammifero dalle fattezze simile all'ermellino. Gli scienziati suppongono che la trasmissione è avvenuta a causa del fatto che in Cina la civetta viene cacciata come cibo.

In conclusione, tale scoperta ha permesso uno studio più approfondito del virus, oltre ad aver ridotto la sua diffusione.

1

CAPITOLO 1: CONCETTI BASE DI BIOLOGIA

La bioinformatica è una materia che tratta tanto l'informatica quanto la biologia, pertanto è necessario illustrarne gli argomenti più importanti, che verranno spiegati in modo funzionale allo scopo della presente tesi.
La *biologia* è la scienza che studia la vita, dagli attori che ne fanno parte fino ai processi in cui essi sono coinvolti [12]. Poiché la vita sulla terra si estende dalle profondità del mare fino alla biosfera, si è reso necessario organizzarla in differenti ordini di grandezza. L'atomo è l'unità elementare che costituisce tutta la materia. Insiemi di atomi formano le molecole che, a loro volta combinandosi, formano le macromolecole. Insieme sono i costituenti delle cellule, le più piccole strutture classificabili come organismi viventi.
Ci sono quattro tipi di macromolecole essenziali per tutte le forme di vita:

- *Polisaccaridi*: macromolecole formate da aggregazioni di monosaccaridi, tra cui il fruttosio, il glucosio e così via. Sono riserve di energia pronta.
- *Proteine*: svolgono una vasta gamma di funzioni all'interno degli organismi viventi, permettendo le reazioni metaboliche, la replicazione del DNA, la risposta agli stimoli e così via.
- *Lipidi*: chiamati anche grassi, sono le riserve energetiche di deposito.
- *Acidi nucleici*: DNA e RNA, contengono e trasportano l'informazione genetica.

Le sezioni successive del capitolo sono dedicate al DNA, RNA e proteine.

1.1 DNA

Il *DNA* o *acido desossiribonucleico* è una macromolecola contenente il patrimonio genetico[1] degli esseri viventi [12], dunque ne detiene l'informazione ereditaria [8].
Porzioni specifiche di DNA contengono determinate informazioni, ad esempio il colore degli occhi, dei capelli e così via. Queste prendono il nome di *gene* [9].
Di seguito viene illustrata un'immagine del DNA, insieme ad una breve descrizione.

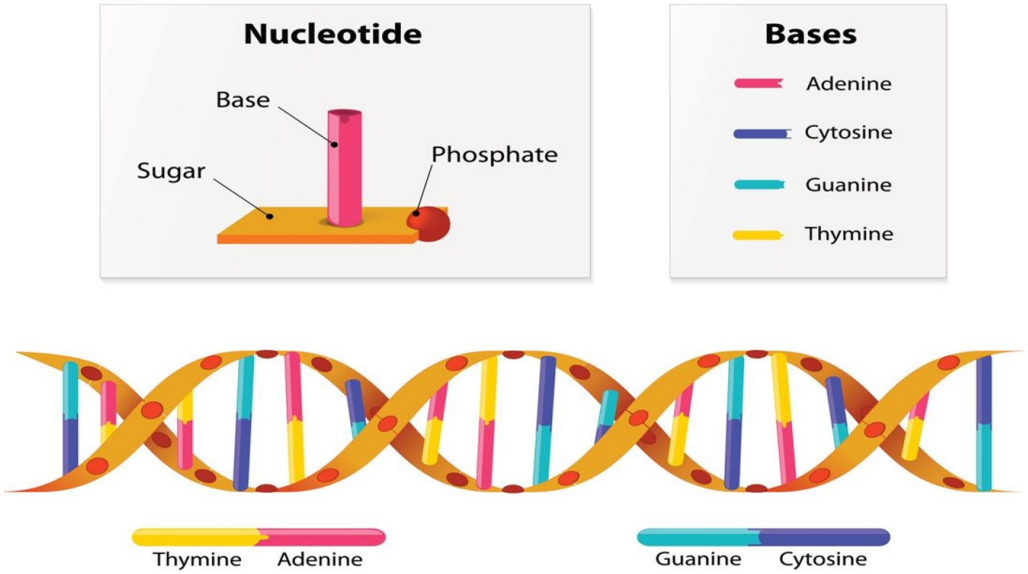

Figure 1: La struttura del DNA e del nucleotide.

La struttura è caratterizzata da una doppia elica di lunghezza variabile, dove ciascun filamento è formato da una sequenza di molecole chiamate *desossiribonucleotidi*.
Un desossiribonucleotide è composto da una molecola di zucchero, un gruppo fosfato ed una base azotata. Di quest'ultima, ne esistono quattro tipi:

- *adenina* (A);
- *timina* (T);
- *guanina* (G);

1 Il patrimonio genetico contiene tutte le informazioni genetiche di un organismo.

- *citosina* (C).

Una proprietà importante delle basi azotate è che sono biunivocamente legate tra loro: l'Adenina si può legare solo con la Timina (A-T), mentre la Guanina con la Citosina (G-C). Questo significa che i filamenti sono complementari e quindi se conosciamo la sequenza di basi di un filamento di DNA sappiamo anche la sequenza di quello complementare.

1.2 RNA

L'*RNA*, ovvero *acido ribonucleico*, è una macromolecola caratterizzata da una struttura a singolo filamento composta da una sequenza di lunghezza variabile di *ribonucleotidi*.
I ribonucleotidi si differenziano rispetto ai desossiribonucleotidi per una diversa molecola di zucchero e per la presenza della base azotata Uracile (U) che sostituisce la Timina.
Un tipo di RNA importante è l'*RNA messaggero* (mRNA), che trasporta l'informazione genetica contenuta nel DNA in una regione cellulare (citoplasma) in cui avviene la sintesi delle proteine[2].

1.3 PROTEINE

Le proteine sono le fondamenta di un organismo, infatti determinano la struttura e le funzioni delle cellule, ad esempio le cellule del cervello differiscono da quelle dei muscoli principalmente perché usano tipi diversi di proteine. La loro struttura è composta da sequenze di *aminoacidi* legati tra loro.
Sebbene esistano oltre cinquecento aminoacidi in natura, solo venti sono codificati dal codice genetico umano e pertanto utilizzati per la sintesi proteica.
La conversione dell'informazione genetica dal DNA in proteina, avviene in due processi, di seguito elencati:

1. *trascrizione*: viene prodotto l'RNA messaggero che trasporta l'informazione nel citoplasma dove avverrà la traduzione;

2. *traduzione*: l'informazione contenuta all'interno del mRNA viene convertita in proteine.

2 La sintesi proteica è il processo attraverso il quale vengono prodotte nuove proteine.

2

CAPITOLO 2: LA BIOINFORMATICA

Per molti anni l'informatica è stata una scienza a sé stante, tuttavia negli ultimi decenni, grazie al progresso scientifico e tecnologico, sono nate nuove discipline chiamate genericamente **X-Informatics**. Queste sono il risultato dell'incontro tra l'informatica ed altre scienze di base (quali la biologia, la chimica, l'astronomia, la geologia etc) e tra queste citiamo la bioinformatica, la chemioinformatica, l'astroinformatica, la geoinformatica e così via. Anche se queste discipline sono diverse tra loro, condividono gli stessi obiettivi:

- elaborazione ed estrazione delle informazioni;
- integrazione dei dati ottenuti tra sorgenti eterogenee;
- utilizzo trasparente ed efficiente dei dati in base al contesto scientifico, dalla raccolta, all'analisi, fino alla catalogazione;
- fornire supporto decisionale per l'utente, riducendo così i possibili errori e facilitando l'analisi dei risultati.

Tra tutte queste discipline risulta di particolare importanza la bioinformatica.

2.1 CHE COSA È LA BIOINFORMATICA?

Non esiste un unico modo con cui definire la bioinformatica, infatti è possibile trovare definizioni diverse tra loro in quanto i professionisti non sempre concordano sulla portata del suo uso, sia nel campo della biologia che dell'informatica. Tuttavia una possibile definizione è la seguente:

La bioinformatica è un campo multidisciplinare della scienza che coinvolge la genetica, la biologia molecolare, l'informatica, la matematica e la statistica, rivolta a studiare sistemi biologici utilizzando metodi e modelli informatici e computazionali [15] [27] [42].

Tra i vari obiettivi precedentemente elencati, va aggiunto quello che risulta essere l'obiettivo principale di questa disciplina, ovvero aumentare la conoscenza di tutti quei processi ed attori presenti in natura.
A prescindere dal problema in questione, è possibile individuare un approccio standard, suddiviso in quattro step:

1. analisi del problema da affrontare;
2. raccolta ed analisi di dati statistici ottenuti a fronte di dati biologici in input;
3. creazione di modelli ed uso di strumenti matematici che possano essere applicati al problema in esame, al fine di sviluppare un algoritmo;
4. creazione, valutazione e test dell'algoritmo risolutivo del problema;

Una parte fondamentale della bioinformatica consiste in esperimenti che generano dati ad alto throughput (high-throughput data), tra cui la misurazione dei modelli di espressione genica oppure la determinazione della sequenza nucleotidica nel caso del DNA e RNA e aminoacidica nel caso delle proteine. Per *high-throughput data* si intendono tutti quei dati biologici ottenuti tramite tecniche automatizzate e quindi non ottenibili attraverso metodi convenzionali [20]. Il mining di questi dati può portare a nuove scoperte scientifiche non solo in campo biologico, ma anche medico, sia nel breve che nel lungo periodo. Ad esempio, nel breve periodo, grazie all'estrazione dei dati ottenuti dal *progetto genoma umano*[1], sono stati scoperti nuovi geni legati alle malattie e nuovi bersagli molecolari[2]. Nel lungo periodo sarà possibile scoprire eventuali reazioni avverse ai farmaci che sono differenti da individuo ad individuo, al punto tale che, grazie all'informazione genetica ottenuta attraverso strumenti informatici, sarà possibile personalizzare l'uso di farmaci, portando ad avere una terapia individuale di maggiore efficacia, riducendo o addirittura eliminando possibili effetti collaterali.

1 Il progetto genoma umano (Human Genome Project) è stato uno dei più grandi progetti scientifici degli ultimi anni. L'obiettivo era quello di ottenere la sequenza del genoma umano (e quindi il suo intero DNA) e identificare i geni contenuti in esso. Il progetto è cominciato nel 1990, per poi essere completato nel 2003 ed ulteriori ricerche sono ancora in corso.

2 Il bersaglio molecolare è una qualsiasi proteina o enzima su cui si può intervenire per modificare il decorso di una malattia.

2.2 STORIA

Fino agli anni '50 il DNA era ancora una scoperta a "metà", il World Wide Web non era ancora nato e nessuno sentiva l'esigenza di dover creare algoritmi per analizzare e memorizzare i dati biologici. Tuttavia le cose cambiarono dal 1953 in poi, con la scoperta della struttura a doppia elica del DNA. Da quel periodo in poi si sono susseguite una serie di scoperte scientifiche che hanno reso la biologia una scienza molto orientata ai dati. La bioinformatica nasce verso la fine degli anni '70, con la scoperta delle prime sequenze nucleotidiche del DNA nasce l'esigenza di poter archiviare i dati e consultarli quando necessario. Da quegli anni in poi la bioinformatica è cresciuta insieme alla biologia e tuttora è una scienza in continua evoluzione.
È possibile trovare altre due date importanti nella storia della bioinformatica, di seguito elencate.

- *Anno 1990*: data di creazione del *Basic Local Alignment Search Tool (BLAST)* [5] ovvero un algoritmo che permette il confronto tra sequenze nucleotidiche e aminoacidiche.
- *Anno 2003*: completamento del *progetto Genoma Umano*, che ha permesso la scoperta dell'intero patrimonio genetico dell'essere umano.

2.3 AREE DI RICERCA

Data la natura eterogenea dei dati biologici, la bioinformatica comprende un vasto numero di aree di ricerca in continua crescita. Di seguito verranno elencate le principali, assieme ai relativi algoritmi più importanti. Sarà possibile notare che alcuni degli algoritmi presentati vengono usati in aree di ricerca diverse, grazie alla loro scalabilità.

2.3.1 *Analisi dei genomi*

Uno dei principali focus della bioinformatica riguarda l'analisi dei genomi[3] degli organismi il cui sequenziamento[4] è già stato completato, dal moscerino della frutta fino all'essere umano. Questa area di ricerca riguarda

3 Per genoma si intende l'intero materiale genetico di un organismo, composto da DNA o RNA.

4 Il sequenziamento è un processo mediante il quale viene determinata la struttura primaria delle macromolecole del DNA, RNA (sequenze di nucleotidi) e proteine(sequenze di aminoacidi).

anche la genomica, ovvero quella disciplina che studia la struttura, il contenuto, la funzione e l'evoluzione del genoma.
Perché analizzare i genomi? Un gene viene sequenziato per conoscere la sua funzione ed eventualmente per poterla modificare. La conoscenza dell'intero genoma di un organismo fornisce le sequenze di tutti i suoi geni, permettendo così di identificare e manipolare quelli che influenzano il metabolismo, lo sviluppo cellulare e i processi patologici negli esseri umani, animali e nelle piante.
L'obiettivo dell'analisi dei genomi è di identificare e modificare i geni che abbiano una particolare funzione biologica attraverso strumenti computazionali, ovvero quelli che permettono la risoluzione di problemi altrimenti inaccessibili con i tempi e le modalità umane.

2.3.2 *Analisi di sequenze*

L'analisi delle sequenze di DNA, RNA o proteine è un processo mediante il quale tali macromolecole vengono sottoposte a dei metodi analitici al fine di capirne la struttura e le funzionalità.
Di particolare importanza risulta lo studio delle sequenze di DNA[5] che possono essere memorizzate in un computer attraverso una vasta varietà di metodi. La memorizzazione avviene attraverso l'uso di caratteristiche identificative di una determinata sequenza di DNA, ad esempio dando un nome al gene o indicandone la fonte, dopodiché vengono salvate all'interno di database che prendono il nome di *database biologici* (vedere sottosezione 2.3.10).
Grazie all'analisi delle sequenze genetiche, è possibile individuare le mutazioni di geni alla base di potenziali malattie.
Una volta estratto il DNA, vengono create migliaia e migliaia di copie di un singolo frammento e successivamente inserite in macchinari chiamati *DNA Sequencer*. Queste ultime svolgono il sequenziamento del DNA in modo automatico. Infine i dati vengono raccolti ed analizzati.
Tra i vari algoritmi utilizzati in questa area di ricerca risultano di particolare importanza gli *algoritmi di Clustering*, il cui obiettivo è quello di raggruppare i dati delle sequenze in insiemi (cluster) in modo veloce e preciso in base a determinati criteri, affinché gli elementi simili tra di loro siano nello stesso cluster mentre quelli differenti risiedono in altri. I

5 Il sequenziamento del DNA consiste nel determinare la sequenza di nucleotidi all'interno di un suo frammento.

principali algoritmi di Clustering nella bioinformatica sono il *Clustering gerarchico* e l'*algoritmo k-Means*.

2.3.3 *Analisi dell'espressione genica*

Quando un gene è attivo, si intende dire che è "espresso", ovvero che è stato prima trascritto in una copia di mRNA (RNA messaggero) e poi tradotto in proteina, con questo concetto si intende *espressione genica*. L'analisi dell'espressione genica quantifica l'attività dell'espressione di migliaia di geni simultaneamente, per capire in quali condizioni alcuni di questi risultano attivi ed altri no.
In questa area di ricerca gli algoritmi di data mining e di Clustering giocano un ruolo di fondamentale importanza, in quanto i primi consentono di estrarre le informazioni mentre i secondi permettono di classificarle in gruppi (in particolar modo il Clustering gerarchico e l'algoritmo K-means).
Ma perché l'analisi dell'espressione genica è importante? Le motivazioni sono principalmente due. Prima di tutto, se l'espressione di un gene non conosciuto è simile a quella di un gene noto, è possibile ipotizzare che essi abbiano funzioni simili o che siano coinvolti nello stesso meccanismo biologico, portando a nuove scoperte scientifiche sui geni. In secondo luogo, è importante anche nel settore della biomedicina, predicendo eventuali metastasi tumorali.

2.3.4 *Analisi ed elaborazione di bioimmagini*

L'analisi ed elaborazione delle bioimmagini consiste nell'usare metodi informatici per acquisire, analizzare, fare data mining di immagini ottenute al microscopio, con l'obiettivo di risolvere problemi di natura biologica e medica. Questa area di ricerca si basa principalmente sul machine learning, in particolar modo sul *pattern recognition*, ovvero lo studio di come le macchine possano imparare a distinguere vari modelli e a prendere delle decisioni in base a specifici pattern, ponendosi come obiettivo principale, nel caso della bioinformatica, la classificazione di organismi.

2.3.5 *Filogenetica*

La filogenetica è quel campo della biologia evolutiva[6] che studia le relazioni evolutive tra le entità attraverso la costruzione di alberi filogenetici. Tradizionalmente si basava sul confronto tra le caratteristiche fenotipiche degli organismi, oggi invece si usano i dati ottenuti tramite sequenziamento genico, permettendo la costruzione di tali alberi in modo estremamente accurato. Questi verranno spiegati esaustivamente nel capitolo successivo, in quanto oggetto principale della presente tesi.
Le applicazioni della filogenetica sono molteplici, elencate di seguito.

- *Bioinformatica e computing*: gli alberi risultano una struttura dati di fondamentale importanza nel campo dell'informatica e degli algoritmi, infatti, molti di questi sviluppati per la filogenetica sono stati successivamente utilizzati anche in altri settori.
- *Classificazione*: fornisce dei metodi di classificazione degli organismi viventi in modo accurato.
- *Medicina forense*: può essere usata per valutare delle prove di DNA in casi giudiziari.
- *Identificazione dell'origine evolutiva degli agenti patogeni*: è possibile studiare ancora più a fondo gli agenti patogeni[7], prevenendo eventuali epidemie. Infatti, scoprire a quale specie vivente è collegato un determinato agente patogeno fornisce delle informazioni per scoprire quale potrebbe essere una eventuale forma di trasmissione.
- *Conservazione delle specie*: contribuisce ad impedire l'estinzione di specie animali e vegetali.

2.3.6 *Bioinformatica strutturale*

Le macromolecole, tra cui DNA, RNA e proteine, svolgono la loro funzione all'interno dei sistemi biologici grazie alla loro conformazione tridimensionale, cioé una forma particolare che assumono nello spazio: questo è particolarmente vero per le proteine che senza specifici ripiegamenti su sé stesse sono prive di qualsiasi funzione. Tuttavia conoscere tale struttura non è affatto facile, basti pensare che in natura esistono

6 La biologia evolutiva si occupa dello studio delle origini ed evoluzione delle specie.
7 Gli agenti patogeni sono i virus,i batteri, etc...

20 diversi tipi di aminoacidi e che se prendiamo una proteina composta da una sequenza di 70 di questi, è possibile ottenere $20^{70} (= 1.180591620717411303424 \times 10^{91})$ strutture diverse (anche se la natura non ne ha selezionate così tante!). Ed è qui che entra in gioco la bioinformatica strutturale, ovvero quella area di ricerca che si pone l'obiettivo di analizzare e ricostruire, tramite algoritmi, la struttura tridimensionale delle macromolecole.
Grazie a questa disciplina è possibile conoscere le interazioni fra macromolecole, nuovi dati biologici e predire[8] la loro struttura tridimensionale, permettendo quindi di conoscerne le funzionalità, in particolar modo per le proteine. Proprio quest'ultimo obiettivo risulta di particolare importanza non solo per la bioinformatica ma in generale per la biologia stessa, oltre ad essere tutt'ora una grande sfida per la scienza.
In passato sono stati utilizzati vari approcci di tipo computazionale, tra cui gli algoritmi evolutivi, tuttavia non risultavano particolarmente efficaci per questo tipo di problema, al contrario invece degli algoritmi genetici.
Gli *algoritmi genetici* sono metodi complessi che hanno il compito di risolvere problemi basati sulla selezione naturale: data una popolazione in input, ad ogni iterazione vengono scelti casualmente degli individui, chiamati genitori, che verranno usati per creare altri individui, chiamati figli, che a loro volta verranno utilizzati nella iterazione successiva, di modo che con il passare delle generazioni si raggiunge la soluzione ottima.
Questi algoritmi risultano particolarmente efficienti nella bioinformatica strutturale per due motivi, il primo è che riescono a risolvere problemi complessi velocemente, sfruttando la parallelizzazione automatica[9] ed il secondo che sono particolarmente ottimizzati per la ricerca genetica.
Da citare *Proteine Data Bank* [36], un vero e proprio archivio di acidi nucleici e proteine visualizzabili in 3-D.

2.3.7 *Genetica delle popolazioni*

Prima di poter definire che cosa è la genetica delle popolazioni, è necessario introdurre alcuni concetti, di seguito elencati.

8 Con predizione si intende la conoscenza di ogni atomo di cui è composta la macromolecola nelle tre dimensioni.

9 La parallelizzazione è un processo mediante il quale invece di eseguire un task alla volta, viene spezzettato in più "sotto-task" indipendenti, che quindi possono essere eseguiti in contemporanea.

- *Popolazione*: è un gruppo di organismi che vivono nello stesso luogo e che condividono determinate proprietà biologiche, pertanto sono della stessa specie.
- *Alleli*: sono le diverse sequenze possibili per un gene, pertanto la loro combinazione determina il suo carattere ereditario (colore degli capelli, degli occhi, ecc...). Ad esempio, nel caso del gene del colore di un fiore, ci può essere un allele per il colore rosso ed un altro per il giallo. Se il primo risulta dominante, allora il fiore sarà di colore rosso.
- *Frequenza genica*: misura la frequenza con cui un allele genico si presenta in una popolazione.
  Preso in considerazione un gene in una popolazione, infatti, è possibile trovare alleli diversi con frequenze diverse.

La genetica delle popolazioni, quindi, si occupa di studiare la frequenza dei geni nelle popolazioni e la loro variazione nello spazio e nel tempo. Poiché in questa area di ricerca vengono coinvolti i processi evolutivi degli organismi, proprio come nella bioinformatica strutturale, gli algoritmi genetici giocano un ruolo chiave (già definiti nella sottosezione 2.3.6).
È possibile dare una sguardo più approfondito a tali algoritmi, individuando cinque fasi fondamentali:

1. *popolazione iniziale*: data una popolazione con un determinato problema, viene scelto un gruppo di geni, rappresentati da stringhe dell'alfabeto;
2. *funzione di fitness*: funzione che associa ad ogni individuo di una popolazione un punteggio che varia in base alla sua abilità nel competere con altri individui;
3. *selezione*: vengono selezionati gli individui con il punteggio di fitness migliore, affinché trasmettano i propri geni alla generazione successiva;
4. *incrocio*: per ogni coppia di genitori viene scelto un punto di scambio tra i loro geni, di modo che i figli erediteranno informazioni genetiche mescolate (crossing over). Questa risulta essere la parte più importante di tali algoritmi;
5. *mutazione*: in alcuni casi ci possono essere delle mutazioni spontanee o indotte dei geni dei figli;

6. *terminazione*: l'algoritmo termina quando non verranno generati più figli, in quanto il problema di partenza è stato risolto.

2.3.8 *Biologia dei sistemi*

Un sistema biologico è una vera e propria rete di entità biologiche connesse tra di loro. Ad esempio, il sistema nervoso di un essere umano è un sistema biologico, composto da un'insieme di entità, ovvero il midollo spinale, i nervi, il cervello ed il cervelletto.
La biologia dei sistemi, quindi, è quella area di ricerca che si occupa di studiare i sistemi biologici attraverso metodi computazionali e modelli matematici e statistici. L'approccio di studio alla materia può essere bottom-up, partendo dai singoli geni fino all'organismo nella sua interezza, oppure viceversa, e quindi top-down.
Le applicazioni della bioinformatica in questa area di ricerca sono molteplici: l'ottenimento di dati ad alto throughput (high-throughput data) attraverso tecniche di analisi statistiche avanzate, e soprattutto la creazione di un linguaggio di Markup per i sistemi biologici, ovvero il *Systems Biology Markup Language* (SBML) [40]. Tale linguaggio è nato con lo scopo di rappresentare e modellare i sistemi biologici (e non solo) attraverso l'uso delle macchine, grazie anche ad un buon numero di tools nati per supportare ed espandere il suo uso, tra cui il framework *Systems Biology Workbench* (SBW) [38]. Il SBW è un insieme di strumenti che permettono di creare, visualizzare e simulare le reti tra le entità che compongono un sistema biologico.

2.3.9 *Data mining*

Data mining, chiamata anche *Knowledge Discovery* è una tecnica di estrazione di informazioni non conosciute e potenzialmente utili a fronte di una grande mole di dati [20]. Con il passare del tempo e con il crescere della quantità di dati biologici, il data mining sta assumendo un ruolo sempre più centrale. Questo si traduce in molte applicazioni, tra cui la ricerca di dati per predire e conoscere la struttura tridimensionale delle proteine, la scoperta delle cause genetiche di una malattia e la facilitazione dell'uso degli algoritmi di clustering di sequenze.
I temi principali del data mining nella bioinformatica sono tre, di seguito elencati.

1. *Data Preprocessing* e *Data Cleaning*: l'eterogeneità delle informazioni biologiche ha reso necessario l'eliminazione dal set di dati da esaminare tutti quelli che risultano formalmente errati (ad esempio "tipo Macromolecola: RNA, Base: Timina") e quelli che risultano inconsistenti, corrotti o mancanti.

2. *Data mining tools*: con il passare degli anni sono stati creati molti strumenti per l'analisi dei dati biologici, quindi, una volta preprocessati e puliti, si possono analizzare proprio grazie a questi strumenti, quali ad esempio *GeneSpring*.

3. *Nuovi metodi di data mining*: nuove ricerche scientifiche richiedono nuovi metodi di estrazione dei dati. Tali metodi devono essere efficienti e scalabili, al fine di poter analizzare al meglio i dati biologici.

2.3.10 *Database biologici*

Poiché la biologia, con il passare degli anni, è diventata una scienza ricca di dati, è nata l'esigenza di catalogarli e memorizzarli. È proprio da ciò che sono nati i database biologici, ovvero quella collezione di informazioni che comprendono sia pubblicazioni scientifiche che dati provenienti da ricerche, di seguito elencati:

- sequenze nucleotidiche (DNA e RNA);
- sequenze di aminoacidi (proteine);
- informazioni su geni (dalla espressione genica al progetto Genoma Umano);
- modelli delle proteine in 3D (ad esempio il Proteine Data Bank [36]).

Poiché questi dati risultano non solo molto diversi tra loro, ma anche complessi da gestire sia nella loro individualità che in relazione agli altri dati, la loro modellazione è un punto cardine per la creazione di database biologici, nei quali è possibile individuare tre concetti fondamentali.

1. *Ordine delle sequenze*: le sequenze nucleotidiche e aminoacidiche possono essere modellate come entità statiche. Questo è dovuto principalmente al fatto che sono delle proprietà interne ad altre entità biologiche e che cambiano molto lentamente con il passare

del tempo (in base all'evoluzione).
Questo concetto è molto importante nella modellazione delle sequenze, in quanto un piccolo cambiamento può avere un grosso impatto sulle entità più grandi.

2. *Processi di input/output*: poiché nella biologia ci sono molti processi che, partendo da un input, restituiscono un output, è necessario identificare ed etichettare il ruolo di queste entità. Ad esempio basti pensare che, partendo con delle sequenze nucleotidiche di DNA, si ottengono delle sequenze di aminoacidiche, grazie ai processi di trascrizione e traduzione.

3. *Relazioni spaziali tra le molecole*: come già detto nella sottosezione "bioinformatica strutturale", la struttura 3D delle macromolecole (in particolar modo le proteine) è alla base per la comprensione delle loro funzioni, pertanto, a livello di database, è necessario definire le relazioni tra le varie entità che compongono una macromolecola.

I database biologici vengono suddivisi in due categorie, ovvero *di primo livello* e *specializzati*. I primi contengono solamente le entità statiche, e quindi le sequenze nucleotidiche e di aminoacidi, mentre i secondi raccolgono informazioni relative alle loro funzioni, alle malattie dovute a mutazioni, pubblicazioni scientifiche, e così via.
Tra i database biologici più famosi è possibile trovare Ensembl [23], National Center for Biotechnology Information (NCBI) [34] ed il già citato Proteine Data Bank [36].

3

CAPITOLO 3: ALBERO EVOLUTIVO

Una delle sfide più importanti della bioinformatica, nonché l'obiettivo principale della filogenetica, è la costruzione degli alberi evolutivi.
Ricordando la definizione di albero, ovvero un grafo non orientato connesso e aciclico [35], *l'albero evolutivo* o *albero filogenetico* è un diagramma che rappresenta le relazioni evolutive tra le varie entità biologiche [1] (animali, piante, virus e così via) [7]. La loro peculiarità consiste nel poter utilizzare dati diversi tra loro, genetici, genomici e morfologici.
Gli alberi evolutivi possono essere suddivisi in due categorie: alberi radicati e non radicati.

3.1 ALBERO RADICATO

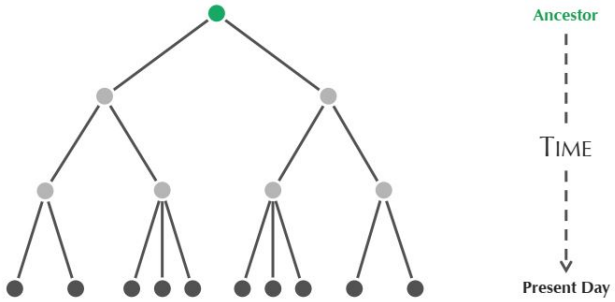

Figure 2: Esempio di albero radicato.

L'albero radicato o *albero con radice* è un albero in cui i nodi (o vertici) rappresentano le entità biologiche, mentre gli archi rappresentano la loro

1 I termini albero evolutivo, albero filogenetico e cladogramma sono spesso usati in modo intercambiabile per descrivere la stessa cosa, ovvero le relazioni evolutive tra entità biologiche. Questo perché l'uso del vocabolario non è sempre coerente nella letteratura scientifica, sebbene il contesto sia lo stesso [18].

evoluzione nel tempo (figura 2) [19]. Esso si sviluppa a partire da un nodo speciale, chiamato *radice* (il vertice verde nella figura) e si estende fino alle foglie. I vertici che hanno grado[2] maggiore di uno, definiti *nodi interni*, sono gli antenati, mentre quelli con grado esattamente uguale ad uno, definite *foglie*, sono le entità attualmente esistenti. La radice, quindi, è l'antenato comune a tutti i vertici dell'albero.
Lo scopo dell'albero non è solo quello di conoscere la radice, ma anche i legami tra le entità.

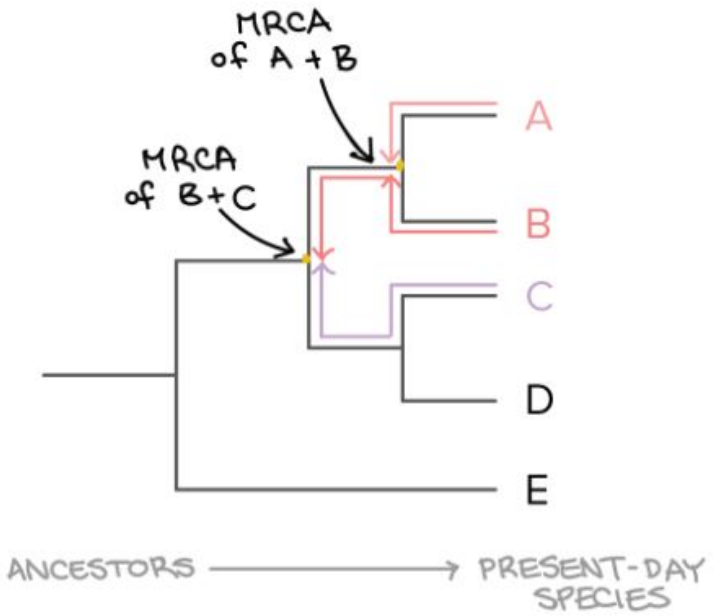

Figure 3: Esempio di albero radicato che mostra le relazioni tra le entità biologiche.

Come mostrato nella figura 3, le entità risultano più legate tra loro rispetto ad altre se hanno l'antenato più vicino, ad esempio A è più legato a B rispetto a C. La sigla "MRCA", infatti, indica il Most Recent Common Ancestor.
Nel caso in cui non sia presente la radice, si parla di albero non radicato.

2 Il grado di un vertice v è dato dal numero degli archi incidenti su v [35].

3.2 ALBERO NON RADICATO

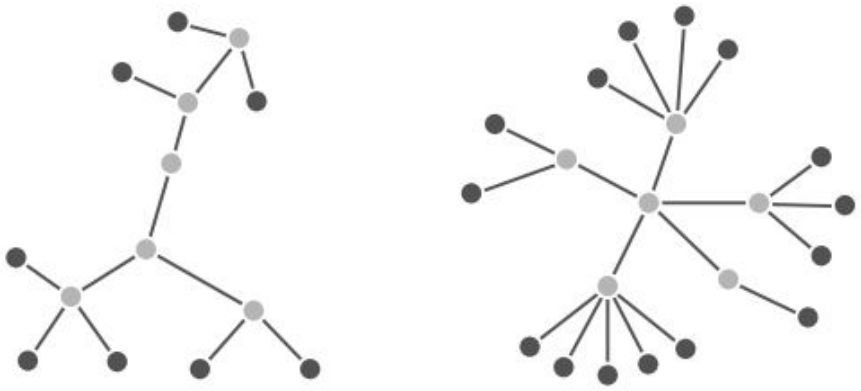

Figure 4: Esempi di alberi non radicati.

L'albero non radicato o *albero senza radice* è un albero in cui i nodi rappresentano le entità biologiche, mentre gli archi rappresentano la loro relazione, pertanto non richiedono la conoscenza della radice (figura 4) [19].
Perché usare gli alberi non radicati invece di quelli con radice? Le motivazioni sono varie.

- Gli alberi senza radice possono essere considerati una generalizzazione di quelli radicati. Questo consente agli scienziati di formulare ipotesi in merito alle relazioni tra le entità in modo più intuitivo.
- Molti degli algoritmi utilizzati costruiscono alberi non radicati e solo successivamente viene trovata la radice, se necessario.

3.3 METODI PER LA COSTRUZIONE DEGLI ALBERI EVOLUTIVI

Gli algoritmi utilizzati per la costruzione degli alberi evolutivi possono essere categorizzati in due metodologie:

- *Metodi basati sulla distanza*: vengono raccolti i dati in una matrice, definita matrice delle distanze. Nel capitolo successivo ne verranno mostrati gli algoritmi.
- *Metodi basati sui caratteri*: vengono usate le sequenze di DNA e di amminoacidi.

4

CAPITOLO 4: ALGORITMI BASATI SULLA DISTANZA

In questo capitolo vengono illustrati i principali algoritmi utilizzati per la costruzione degli alberi evolutivi. L'obiettivo è quello di trovare una soluzione al cosiddetto *problema degli alberi basati sulla distanza*, ma prima è necessario introdurre alcuni concetti, tra cui la matrice delle distanze.

4.1 MATRICE DELLE DISTANZE

Dati due punti, x e y, la *distanza* può essere vista come loro "lontananza" in uno spazio k-dimensionale. Nella fattispecie è una funzione $d(x, y)$ che possiede le seguenti proprietà [20]:

1. *non negatività*:

$$d(x, y) \geqslant 0 \qquad \forall\, x, y \in R^k$$

2. *identità*:

$$d(x, y) = 0 \;\leftrightarrow\; x = y$$

3. *simmetria*:

$$d(x, y) = d(y, x) \qquad \forall\, x, y \in R^k$$

4. *disuguaglianza triangolare*:

$$d(x, y) \leqslant d(x, z) + d(y, z) \qquad \forall\, x, y, z \in R^k$$

Date n unità, calcolando la distanza per ogni coppia di elementi[1] si ottiene una *matrice delle distanze* $n \times n$, definita nel seguente modo [29]:

$$D = \begin{pmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & 0 & d_{23} & \dots & d_{2n} \\ d_{31} & d_{32} & 0 & \dots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & \dots & 0 \end{pmatrix} \qquad \text{dove } d(x_i, x_j) = D_{i,j}$$

Poiché la matrice è costruita a partire dalle distanze, ne eredita le proprietà precedentemente elencate.
Ciascun valore $D_{i,j}$ può assumere significati diversi in base al contesto, ad esempio può indicare il numero di simboli diversi tra i geni i e j nell'allineamento di sequenze di DNA[2], come mostrato nell'esempio sottostante:

SPECIE	ALLINEAMENTO	MATRICE DELLE DISTANZE			
		Umano	Scimpanzé	Foca	Balena
Umano	ATGTAAGACT	0	3	7	5
Scimpanzé	ACGTAGGCCT	3	0	6	4
Foca	TCGAGAGCAC	7	6	0	2
Balena	TCGAAAGCAT	5	4	2	0

Figure 5: Esempio di matrice delle distanze.

Nella figura 5 è possibile notare che la sequenza di DNA della foca risulta molto più simile a quella della balena, in quanto la distanza è 2, rispetto a quella dell'umano, con cui la distanza è 7.

1 Ci sono vari modi per calcolare la distanza tra due elementi, ad esempio attraverso la distanza Euclidea, quella di Manhattan, di Minkowski e così via.

2 L'allineamento è il processo attraverso il quale si misura la similarità tra due o più sequenze.

4.2 PROBLEMA DEGLI ALBERI BASATI SULLA DISTANZA

Gli algoritmi basati sulla distanza utilizzano la matrice delle distanze per costruire gli alberi evolutivi, dove le foglie corrispondono alle entità biologiche presenti nella matrice, mentre i nodi interni rappresentano gli antenati non noti. Per poter calcolare la distanza tra due foglie, e quindi conoscere quanto sono legate tra loro, è necessario associare un valore non negativo (peso) a ciascun arco e definire la lunghezza di un cammino in tale albero come la somma dei pesi degli archi che lo compongono. Si definisce, quindi, la *distanza evolutiva* tra due entità biologiche corrispondenti alle foglie i e j di un albero T come la lunghezza dell'unico cammino che le collega, ed è indicato come $d_{i,j}(T)$ [19]. In altre parole è data dalla somma dei pesi degli archi che ci sono tra i e j. Si dice che un albero T si *adatta* ad una matrice delle distanze D se per ogni coppia di foglie i e j si ha che $D_{i,j} = d_{i,j}(T)$, ovvero l'elemento nella riga i e colonna j è uguale alla lunghezza del cammino che collega le due foglie in T (distanza evolutiva), in tal caso sia la matrice che l'albero vengono definiti *additivi*. Qualora invece non esista un albero che si adatti alla matrice, allora tale matrice è detta *non additiva* [19].
Si riporta di seguito un esempio che mostra un albero che si adatta alla matrice delle distanze mostrata nella sezione precedente.

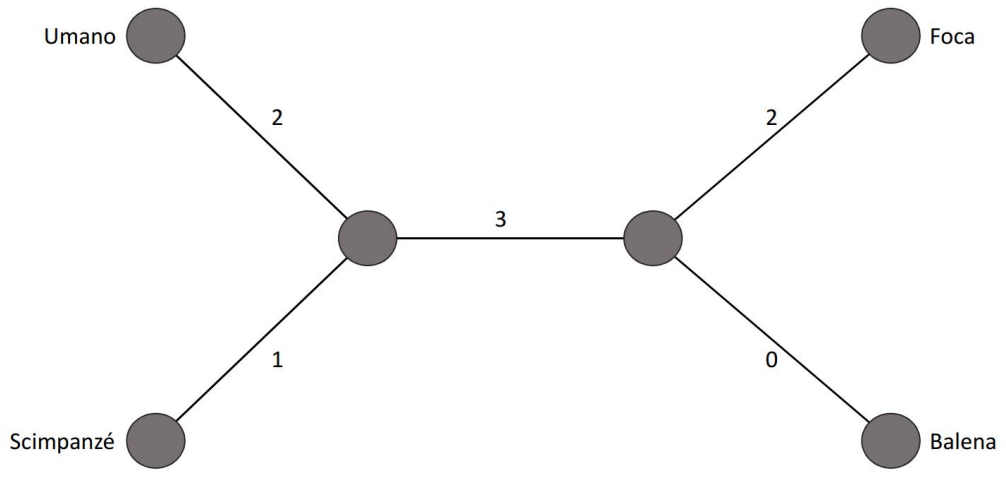

Figure 6: Albero evolutivo senza radice costruito a partire dalla matrice additiva della figura 5.

Ci possono essere più alberi che si adattano ad una matrice, quindi come si può scegliere l'albero giusto? Si nota che quello in figura 6 ha tutti i vertici di grado diverso da due e viene definito *albero semplice*.

Una caratteristica importante è che per ogni matrice delle distanze additiva esiste un unico albero semplice che si adatta alla matrice stessa. Adesso è possibile dare una definizione al problema accennato all'inizio del capitolo, mostrata di seguito.

Problema degli alberi basati sulla distanza:
*Data in **input** una matrice delle distanze additiva restituire in **output** un albero evolutivo semplice.*

4.3 ALGORITMO PER IL PROBLEMA DEGLI ALBERI BASATI SULLA DISTANZA

L'obiettivo è quello di costruire un albero semplice T che si adatti alla matrice delle distanze additiva D.
Si prenda in considerazione la matrice della figura 5, riportata di seguito[3]:

$$D = \begin{array}{c} \text{specie} \\ u \\ s \\ f \\ b \end{array} \begin{array}{c} \begin{array}{cccc} u & s & f & b \end{array} \\ \begin{pmatrix} 0 & 3 & 7 & 5 \\ 3 & 0 & 6 & 4 \\ 7 & 6 & 0 & 2 \\ 5 & 4 & 2 & 0 \end{pmatrix} \end{array}$$

L'idea di base dell'algoritmo è che all'elemento più piccolo della matrice corrispondano due foglie *vicine* nel rispettivo albero, ovvero due foglie aventi lo stesso genitore, con $D_{i,j} = d_{i,j}(T)$.
Poiché l'elemento più piccolo della matrice è D_{fb}, il cui valore è 2, si può supporre che f e b siano vicini. Si indica con p il genitore non noto delle due foglie. La situazione è mostrata di seguito.

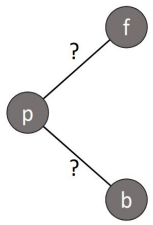

Figure 7: Foglie vicine con un generico genitore p.

Come si può calcolare la distanza tra f e p (d_{fp}) e tra b e p (d_{bp})? Le uniche informazioni a disposizione sono le distanze tra le quattro foglie presenti nella matrice e l'ipotesi che f e b siano vicine. Il passo successivo dell'algoritmo è mostrato di seguito.
Nella figura 8 sono state aggiunte le foglie u ed s, ma il loro arco è tratteggiato in quanto ancora non è possibile sapere come sono collocate nell'albero.
In questo step viene scelta casualmente una foglia tra u ed s e viene sfruttato il valore della sua distanza rispetto a p per poter calcolare d_{fp} e d_{bp} (in questo caso si sceglie u). Per far ciò è quindi necessario scrivere le distanze tra le foglie nel seguente modo:

$$d_{fu} = d_{fp} + d_{up}$$

3 Per brevità si definisce u=umano, s=scimpanzé, f=foca, b=balena.

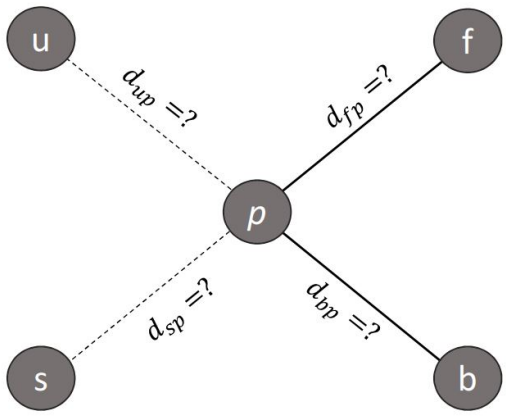

Figure 8: Le quattro foglie u, s, f e d.

$$d_{fb} = d_{fp} + d_{bp}$$

$$d_{bu} = d_{bp} + d_{up}$$

$$d_{up} = \frac{d_{fu} + d_{bu} - d_{fb}}{2} = \frac{(d_{fp} + d_{up}) + (d_{bp} + d_{up}) - (d_{fp} + d_{bp})}{2} =$$

$$= \frac{d_{fp} + d_{up} + d_{bp} + d_{up} - d_{fp} - d_{bp}}{2} = \frac{d_{up} + d_{up}}{2} = \frac{2d_{up}}{2} = d_{up}$$

Perché scrivere d_{up} in quel modo? Perché non si conosce il peso dei nodi interni ma solo quello delle foglie grazie alla matrice di partenza, quindi vengono usate queste informazioni per calcolare la distanza tra f e p e tra b e p nell'albero T. Inoltre D è additiva, questo significa che la distanza tra le foglie in T è uguale alla distanza tra i rispettivi elementi nella matrice, pertanto $d_{fu} = D_{fu}$, $d_{fb} = D_{fb}$ e $d_{bu} = D_{bu}$[4]. Ma allora si può scrivere d_{up} nel seguente modo:

$$d_{up} = \frac{d_{fu} + d_{bu} - d_{fb}}{2} = \frac{D_{fu} + D_{bu} - D_{fb}}{2}$$

A questo punto si è in grado di calcolare la distanza tra f e p e tra b e p, ricavandola dalle uguaglianze trovate poc'anzi. Per prima cosa si calcola d_{fp}:

$$d_{fu} = d_{fp} + d_{up} \rightarrow d_{fp} = d_{fu} - d_{up} = D_{fu} - D_{up} =$$

$$= D_{fu} - \frac{D_{fu} + D_{bu} - D_{fb}}{2} = \frac{2D_{fu} - D_{fu} - D_{bu} + D_{fb}}{2} =$$

4 Si ricorda che d_{fu} è la distanza tra il nodo f ed il nodo u nell'albero T, mentre D_{fu} è l'elemento in posizione $f - u$ in D.

$$= \frac{D_{fu} + D_{fb} - D_{bu}}{2} = \frac{7+2-5}{2} = 2$$

Quindi $d_{fp} = 2$. In modo analogo si calcola d_{bp}:

$$d_{bu} = d_{bp} + d_{up} \rightarrow d_{bp} = d_{bu} - d_{up} = D_{bu} - D_{up} =$$

$$= D_{bu} - \frac{D_{fu} + D_{bu} - D_{fb}}{2} = \frac{2D_{bu} - D_{fu} - D_{bu} + D_{fb}}{2} =$$

$$\frac{D_{bu} + D_{fb} - D_{fu}}{2} = \frac{5+2-7}{2} = 0$$

Quindi $d_{bp} = 0$. Il risultato di questo passo dell'algoritmo è il seguente albero:

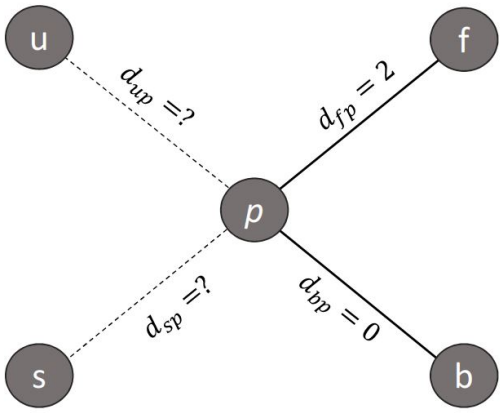

Figure 9: Albero con le distanze d_{fp} e d_{bp}.

Il prossimo step consiste nel trovare la distanza tra u e p e tra s e p: dalla matrice D sappiamo che $D_{fu} = d_{fu} = 7$ e dai calcoli precedenti che $d_{fp} = 2$, quindi $d_{up} = d_{fu} - d_{fp} = 7 - 2 = 5$. Stesso ragionamento anche per d_{sp}, ovvero $d_{sp} = d_{bs} - d_{bp} = 4 - 0 = 4$.
L'albero corrispondente è:

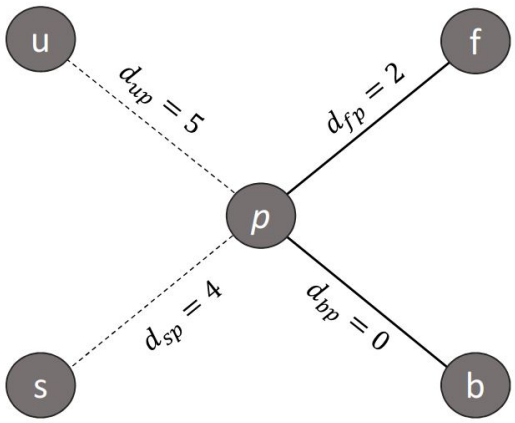

Figure 10: Albero con le distanze d_{fp}, d_{bp}, d_{up} e d_{sp}.

Adesso è necessario fare due modifiche alla matrice D, la prima consiste nel mettere una riga e colonna in più, definite p, che rappresentano le distanze appena trovate, ottenendo quindi la seguente matrice:

$$D = \begin{array}{c} \text{specie} \\ u \\ s \\ f \\ b \\ p \end{array} \begin{array}{c} \begin{array}{ccccc} u & s & f & b & p \end{array} \\ \begin{pmatrix} 0 & 3 & 7 & 5 & 5 \\ 3 & 0 & 6 & 4 & 4 \\ 7 & 6 & 0 & 2 & 2 \\ 5 & 4 & 2 & 0 & 0 \\ 5 & 4 & 2 & 0 & 0 \end{pmatrix} \end{array}$$

la seconda invece consiste nel togliere le righe e colonne appartenenti a f e b, in quanto i rispettivi nodi sono già aggiunti all'albero, pertanto non più utili. Quindi:

$$D = \begin{array}{c} \text{specie} \\ u \\ s \\ p \end{array} \begin{array}{c} \begin{array}{ccc} u & s & p \end{array} \\ \begin{pmatrix} 0 & 3 & 5 \\ 3 & 0 & 4 \\ 5 & 4 & 0 \end{pmatrix} \end{array}$$

Anche se adesso si conoscono le distanze di tutte le foglie con il genitore p, rimane comunque da capire se u e s abbiano altri genitori. Si ricorda, infatti, che i loro archi sono stati tratteggiati in quanto ancora non si conosce la loro collocazione nell'albero. Per risolvere questo problema basta applicare ricorsivamente i passi precedenti, quindi si individua l'elemento più piccolo della matrice (d_{su}) e si suppone che s e u siano vicini e quindi che abbiano un genitore non noto k, come mostrato in figura 11.

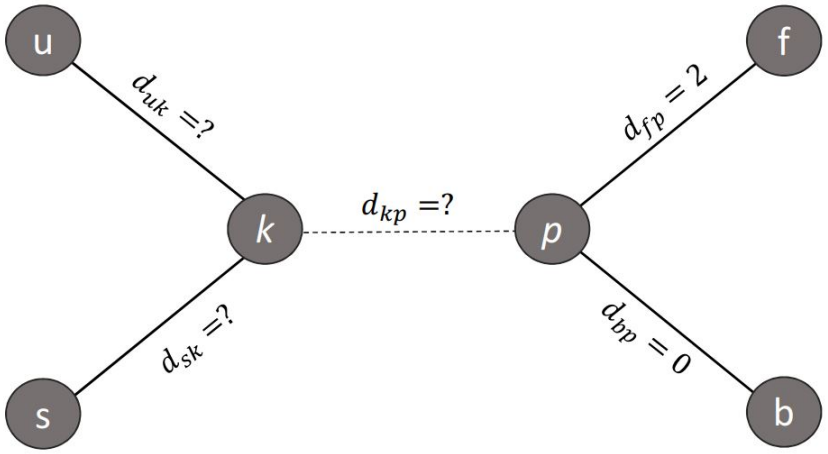

Figure 11: Albero con il nuovo genitore non noto k.

Viene scelto il nodo p e sfruttato il valore della sua distanza rispetto ad u e s per poter calcolare d_{uk} e d_{sk}. Adattando le formule di pagina 31 con i dati attuali, si ottiene:

$$d_{pk} = \frac{d_{up} + d_{sp} - d_{us}}{2} = \frac{(d_{uk} + d_{pk}) + (d_{sk} + d_{pk}) - (d_{uk} + d_{sk})}{2} =$$

$$= \frac{2d_{pk}}{2} = d_{pk}$$

Adesso si può calcolare d_{uk}:

$$d_{up} = d_{uk} + d_{pk} \rightarrow d_{uk} = D_{up} - D_{pk} = D_{up} - \frac{D_{up} + D_{sp} - D_{us}}{2} =$$

$$= \frac{D_{up} + D_{us} - D_{sp}}{2} = 2$$

E d_{sk}:

$$d_{sp} = d_{sk} + d_{pk} \rightarrow d_{sk} = D_{sp} - D_{pk} = D_{sp} - \frac{D_{up} + D_{sp} - D_{us}}{2} =$$

$$= \frac{D_{sp} + D_{us} - D_{up}}{2} = 1$$

Applicando le distanze trovate all'albero della figura 11, si ottiene:

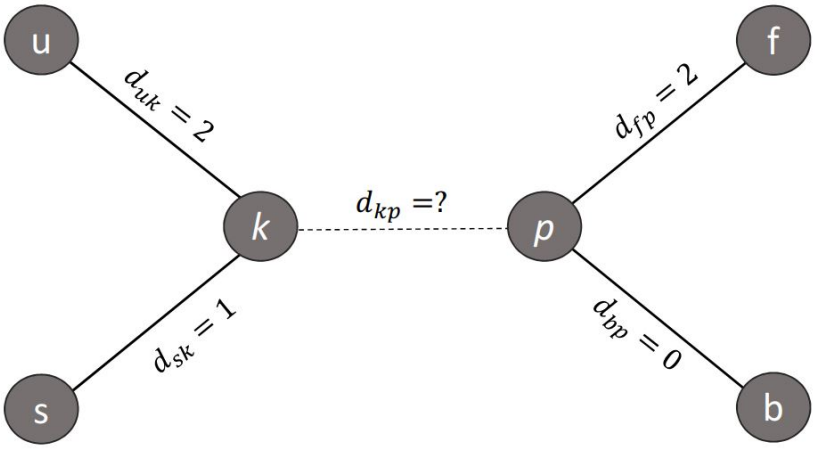

Figure 12: Albero con le distanze d_{uk} e d_{sk} calcolate.

Dopo aver trovato ricorsivamente la distanza delle foglie dai propri genitori e di conseguenza aver aggiornato la matrice, rimane un ultimo passo da completare, ovvero calcolare la distanza tra k e p: in precedenza si era calcolata la distanza tra u e p e tra s e p, quindi basta sottrarre tali valori a quelli trovati adesso e troviamo d_{kp}.

$$d_{kp} = d_{sp} - d_{sk} = 4 - 1 = d_{up} - d_{uk} = 5 - 2 = 3$$

L'albero finale che si ottiene è il seguente:

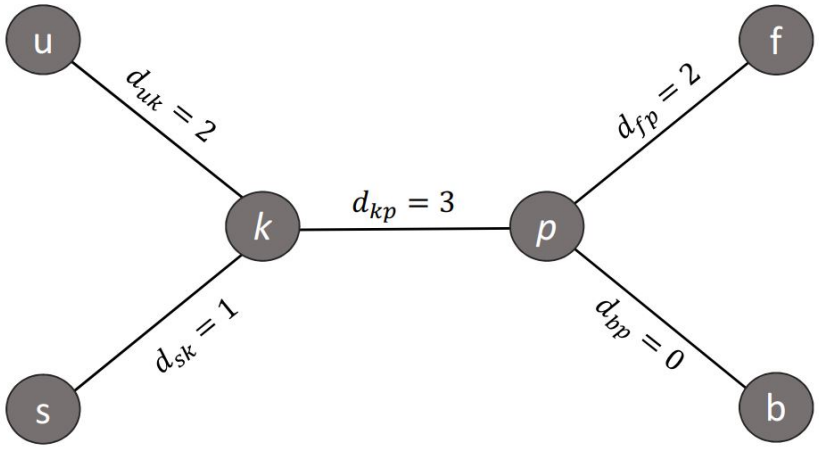

Figure 13: Albero evolutivo semplice con tutte le distanze calcolate.

L'algoritmo è terminato. Tuttavia vi è una criticità che viene mostrata nella sezione successiva.

4.3.1 *Complessità temporale*

Per calcolare la complessità nel tempo dell'algoritmo possiamo suddividerlo in tre step, di seguito elencati.

1. *Step 1*: Trovare il minimo in una matrice di dimensione $n \times n$.
   Si suppone l'utilizzo di un algoritmo di ricerca lineare, la cui complessità nel caso pessimo, dato in input un vettore di lunghezza n, è pari a $O(n)$. Poiché questa operazione deve essere effettuata per tutte le n righe, otteniamo che:

   $$T(step1) = O(n) \times O(n) = O(n^2)$$

2. *Step 2*: Trovare il genitore per ogni coppia di foglie e calcolare la distanza di tutte le n foglie rispetto al genitore stesso.
   Questa operazione deve essere fatta n volte, quindi $O(n)$. In seguito dovrà essere aggiornata la matrice; per semplicità si suppone che il costo rimanga invariato, quindi $O(1)$.

   $$T(step2) = O(n) + O(1) \simeq O(n)$$

3. *Step 3*: Calcolare le distanze tra i genitori trovati nello step precedente.
   Poiché per ogni coppia di foglie esiste un genitore, allora il costo sarà:

   $$T(step3) = O(n/2)$$

Adesso è sufficiente sommare le tre complessità e si trova quella totale dell'algoritmo:

$$T(totale) = T(step1) + T(step2) + T(step3) =$$

$$= O(n^2) + O(n) + O(n/2) \simeq O(n^2)$$

4.4 ALBERO ADDITIVO

L'algoritmo mostrato nella sezione precedente presenta una criticità, infatti riesce a risolvere il problema degli *alberi basati sulla distanza* solamente se l'elemento più piccolo della matrice D corrisponde a due foglie *vicine* nell'albero T. Ma questo non è necessariamente vero, infatti ci sono matrici delle distanze che, seppur additive (e quindi $\forall i, j \in V, D_{ij} = d_{ij}(T)$), non possiedono come elemento più piccolo una coppia di foglie con lo stesso genitore. Pertanto si deve approcciare il problema in modo diverso: invece di cercare le foglie vicine in un albero (come visto nella sezione precedente), l'idea di base dell'algoritmo è quella di costruirlo aggiungendole una alla volta. Per far ciò è necessario conoscere il peso degli archi che collegano le foglie ai rispettivi genitori. Questi prendono il nome di *arti*. Tutti gli altri sono definiti *archi interni* [20]. Tale operazione andrà effettuata in termini di matrice delle distanze D, dato che non si conosce l'albero T fino a che l'algoritmo non è terminato.
Il primo problema che si pone è quello di calcolare il peso degli arti: data una foglia j, si denota con *limbweight(j)* il peso dell'arco che collega j al suo genitore [29].

Teorema del peso degli arti:
Data una matrice delle distanze additiva D *ed una foglia* j*, limbweight(j) è uguale al valore minimo di* $\frac{D_{j,i}+D_{j,k}-D_{i,k}}{2}$ *tra tutte le foglie* i *e* k[5] *[19] .*

A titolo esemplificativo si riporta l'applicazione del teorema alla seguente matrice[6]:

$$D = \begin{array}{c} \text{specie} \\ u \\ s \\ f \\ b \end{array} \begin{array}{c} \begin{array}{cccc} u & s & f & b \end{array} \\ \begin{pmatrix} 0 & 3 & 7 & 5 \\ 3 & 0 & 6 & 4 \\ 7 & 6 & 0 & 2 \\ 5 & 4 & 2 & 0 \end{pmatrix} \end{array}$$

Supponendo di non conoscere quali sono le foglie vicine, si vuole calcolare il peso dell'arto di f, quindi:

$$\mathtt{limbweight}(f) = \frac{D_{f,i} + D_{f,k} - D_{i,k}}{2}$$

5 Si omette la dimostrazione, che è consultabile al capitolo 7 del libro *Bioinformatics Algorithms, An Active Learning Approach, Vol II* di Compeau P., Pevzner P.

6 Per brevità si definisce f=foca, b=balena, u=umano, s=scimpanzé. Inoltre si ricorda che questi rappresentano le foglie nel rispettivo albero evolutivo T.

Al posto i e k si sostituiscono di volta in volta i valori delle foglie (escluso f), ovvero:

- $i = u$ e $k = s$:

$$limbweight_{us}(f) = \frac{D_{f,u} + D_{f,s} - D_{u,s}}{2} = \frac{7+6-3}{2} = \frac{10}{2} = 5$$

- $i = u$ e $k = b$:

$$limbweight_{ub}(f) = \frac{D_{f,u} + D_{f,b} - D_{u,b}}{2} = \frac{7+2-5}{2} = \frac{4}{2} = 2$$

- $i = s$ e $k = b$:

$$limbweight_{sb}(f) = \frac{D_{f,s} + D_{f,b} - D_{s,b}}{2} = \frac{6+2-4}{2} = \frac{4}{2} = 2$$

Si può concludere dicendo che *limbweight(f)*=2.
Tali operazioni hanno una complessità di $O(n^2)$, in quanto ogni $i-esimo$ elemento della matrice $n \times n$ viene confrontato con tutti gli altri k elementi (con $k > i$), che sono $n-3$ (escluso f, i e sé stesso). Questa operazione viene eseguita tante volte quante sono le foglie da analizzare, ovvero n, quindi $\rightarrow T(limbweight(f)) = O(n-3) \times O(n) \simeq O(n^2)$.
Ora che si è in grado di calcolare il peso degli arti si può illustrare l'algoritmo che risolve la criticità spiegata ad inizio della sezione.
Sia D una matrice delle distanze additiva di dimensione $n \times n$, tale che il suo elemento più piccolo **non** corrisponda a due foglie *vicine* nel rispettivo albero T[7], ovvero:

$$D = \begin{array}{c} \begin{array}{cccc} \text{specie} & f & b & u & s \end{array} \\ \begin{array}{c} f \\ b \\ u \\ s \end{array} \begin{pmatrix} 0 & 13 & 21 & 22 \\ 13 & 0 & 12 & 13 \\ 21 & 12 & 0 & 13 \\ 22 & 13 & 13 & 0 \end{pmatrix} \end{array}$$

7 Per capire se l'elemento più piccolo di una matrice corrisponda ad una coppia di foglie vicine nel rispettivo albero è necessario eseguire l'algoritmo della sezione precedente (sezione 4.3) usando la matrice D in input. In tal caso il peso di alcuni archi sarebbe risultato negativo e quindi non sarebbe possibile costruire T.

L'albero che si intende ottenere è il seguente:

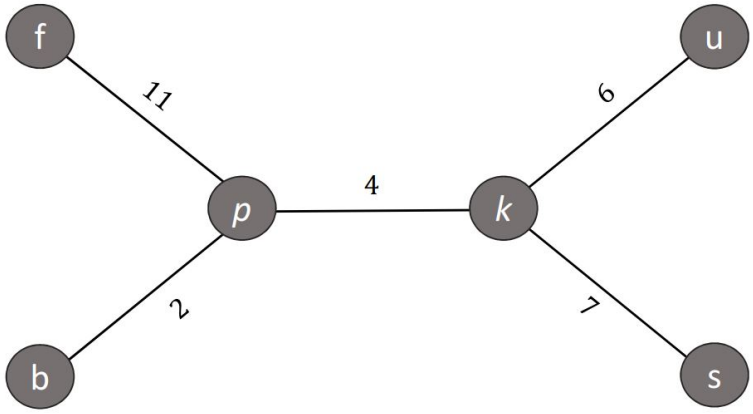

Figure 14: Albero T ottenuto dalla matrice D.

Per prima cosa si sceglie casualmente dalla matrice una foglia b e si calcola *limbweight(b)*:

$$limbweight(b) = \frac{D_{b,i} + D_{b,k} - D_{i,k}}{2}$$

- $i = f$ e $k = u$:

$$limbweight_{fu}(b) = \frac{D_{b,f} + D_{b,u} - D_{f,u}}{2} = \frac{13 + 12 - 21}{2} = \frac{4}{2} = 2$$

- $i = f$ e $k = s$:

$$limbweight_{fs}(b) = \frac{D_{b,f} + D_{b,s} - D_{f,s}}{2} = \frac{13 + 13 - 22}{2} = \frac{4}{2} = 2$$

- $i = u$ e $k = s$:

$$limbweight_{us}(b) = \frac{D_{b,u} + D_{b,s} - D_{u,s}}{2} = \frac{12 + 13 - 13}{2} = \frac{12}{2} = 6$$

Quindi $limbweight(b) = 2$.
Adesso è necessario aggiornare la matrice, sottraendo $limbweight(b)$ a tutti gli elementi lungo la riga e la colonna b (escluso lo zero sulla diagonale). Quindi si ottiene una nuova matrice $D^{spoglia_b}$ che verrà utilizzata successivamente per costruire l'albero.

$$D^{spoglia_b} = \begin{array}{c} \text{specie} \\ f \\ b \\ u \\ s \end{array} \begin{array}{c} \begin{array}{cccc} f & b & u & s \end{array} \\ \begin{pmatrix} 0 & 11 & 21 & 22 \\ 11 & 0 & 10 & 11 \\ 21 & 10 & 0 & 13 \\ 22 & 11 & 13 & 0 \end{pmatrix} \end{array}$$

Poiché la riga e la colonna b non forniscono informazioni aggiuntive, sarebbe inutile tenerle, pertanto si eliminano dalla matrice $D^{spoglia_b}$. Il risultato è il seguente:

$$D^{tagliata_b} = \begin{array}{c} \text{specie} \\ f \\ u \\ s \end{array} \begin{array}{c} \begin{array}{ccc} f & u & s \end{array} \\ \begin{pmatrix} 0 & 21 & 22 \\ 21 & 0 & 13 \\ 22 & 13 & 0 \end{pmatrix} \end{array}$$

Adesso si può ottenere l'albero evolutivo T eseguendo ricorsivamente i seguenti step:

1. Scegliere casualmente una foglia j, calcolare $limbweight(j)$, costruire $D^{spoglia_j}$ e $D^{tagliata_j}$. Ripetere queste operazioni fino a che non si ottiene una matrice $D^{tagliata}$ di dimensioni 2×2.

2. Costruire un albero $T^{tagliato}$ a partire da $D^{tagliata}$ trovata nello step precedente. All'inizio sarà composto solamente da una coppia di foglie ed un arco che le collega.

3. Identificare il punto di $T^{tagliato}$ in cui la foglia j deve essere inserita. Questo passo dell'algoritmo verrà spiegato successivamente.

4. Aggiungere la foglia j ed aggiornare il peso degli archi in $T^{tagliato}$. La ripetizione di questo punto e di quello precedente permettono la costruzione dell'albero di volta in volta fino al suo completamento.

Si applicano i passaggi sopraelencati alla matrice $D^{tagliata_b}$, quindi si sceglie la foglia s e si calcola *limbweight(s)*:

$$limbweight(s) = \frac{D_{s,i} + D_{s,k} - D_{i,k}}{2}$$

- $i = f$ e $k = u$:

$$limbweight(s) = \frac{D_{s,f} + D_{s,u} - D_{f,u}}{2} = \frac{22 + 13 - 21}{2} = \frac{14}{2} = 7$$

Si sottrae $limbweight(s) = 7$ a tutti gli elementi lungo la riga e la colonna s in $D^{tagliata_b}$, ottenendo così:

$$D^{spoglia_s} = \begin{array}{c} \text{specie} \\ f \\ u \\ s \end{array} \begin{array}{c} \begin{array}{ccc} f & u & s \end{array} \\ \begin{pmatrix} 0 & 21 & 15 \\ 21 & 0 & 6 \\ 15 & 6 & 0 \end{pmatrix} \end{array}$$

A questo punto si elimina la foglia s da $D^{spoglia_s}$:

$$D^{tagliata_s} = \begin{array}{c} \\ f \\ u \end{array} \begin{array}{c} \begin{array}{cc} specie & f \quad u \end{array} \\ \begin{pmatrix} 0 & 21 \\ 21 & 0 \end{pmatrix} \end{array}$$

Il risultato è una matrice 2×2, quindi si può cominciare a costruire l'albero:

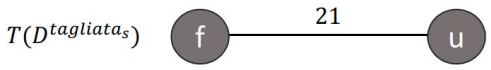

Figure 15: Albero ottenuto dalla matrice $D^{tagliata_s}$.

Lo step successivo è quello di individuare il punto in $T(D^{tagliata_s})$ in cui s deve essere inserito. Per fare ciò si consideri la matrice $D^{spoglia_s}$. Dal *teorema del peso degli arti* si ha che:

$$limbweight(s) = \frac{D^{spoglia_s}_{f,s} + D^{spoglia_s}_{s,u} - D^{spoglia_s}_{f,u}}{2}$$

Ma poiché $D^{spoglia_s}$ è una matrice ottenuta da $D^{tagliata_b}$ togliendo il peso dell'arto di s, allora $limbweight(s) = 0$. Quindi:

$$0 = \frac{D^{spoglia_s}_{f,s} + D^{spoglia_s}_{s,u} - D^{spoglia_s}_{f,u}}{2} \rightarrow D^{spoglia_s}_{f,u} = D^{spoglia_s}_{f,s} + D^{spoglia_s}_{s,u}$$

Questo significa che il punto di attacco di s è lungo l'arco che collega f con u[8], quindi si crea un nodo interno k che rappresenta il genitore di s, si aggiunge la foglia s connettendola proprio a k tramite un arto di peso pari a limbweight(s) = 7 e si aggiorna il peso degli altri archi, quindi:

8 Il punto di attacco può essere anche su un vertice già esistente, in tal caso si connette s a tale vertice.

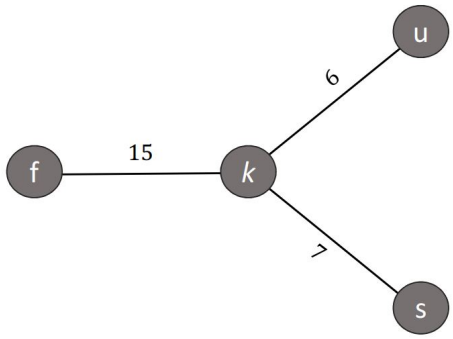

Figure 16: Albero con la nuova foglia s.

$d_{sk} = 7$ è il peso dell'arto di s che è già stato calcolato in precedenza; $d_{uk} = D_{s,u}^{spoglia_s} = 6$ è ottenuto sottraendo $limbweight(s) = 7$ a $D_{s,u}^{tagliata_b} = 13$; $d_{fk} = 15$ è dato da $D_{f,u}^{spoglia_s} - d_{uk} = 21 - 6 = 15$.
Si ripetono gli ultimi step dell'algoritmo fino a che non si ottiene l'albero T, quindi si individua il punto in figura 16 in cui b deve essere inserito.

$$limbweight(b) = \frac{D_{f,b}^{spoglia_b} + D_{b,u}^{spoglia_b} - D_{f,u}^{spoglia_b}}{2} \rightarrow$$

$$\rightarrow limbweight(b) = 0, \, quindi \rightarrow 0 = \frac{D_{f,b}^{spoglia_b} + D_{b,u}^{spoglia_b} - D_{f,u}^{spoglia_b}}{2} \rightarrow$$

$$\rightarrow D_{f,u}^{spoglia_b} = D_{f,b}^{spoglia_b} + D_{b,u}^{spoglia_b}$$

Analogamente al passaggio effettuato poc'anzi, si nota che il punto di attacco di b è lungo l'arco che collega f con u, quindi si crea un nodo interno p che rappresenta il genitore di b, si aggiunge la foglia b connettendola a p tramite un arto di peso pari a $limbweight(b) = 2$ e si aggiorna il peso degli altri archi.

L'albero evolutivo è il seguente:

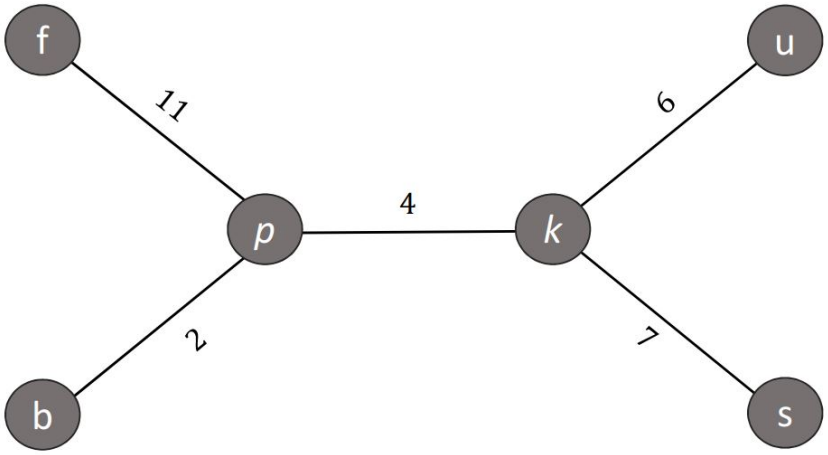

Figure 17: Albero T ottenuto da D.

$d_{bp} = 2$ è il peso dell'arto di b che è già stato calcolato in precedenza; $d_{fp} = D_{f,b}^{spoglia_b} = 11$ è ottenuto sottraendo $limbweight(b) = 2$ a $D_{f,b} = 13$; $d_{pk} = 4$ è dato dalla sottrazione tra d_{fk} (figura 16) e $d_{fp} = D_{f,b}^{spoglia_b}$, quindi $d_{fk} - d_{fp} = 15 - 11 = 4$.
Poiché è stato costruito l'albero nella sua interezza, l'algoritmo è terminato. Tuttavia anche in questo caso vi è una criticità, in quanto riesce a costruire T *se e solo se* la matrice D è additiva, ed in molti casi non è così. Nella sezione successiva viene mostrato un algoritmo che risolve tale criticità.

4.4.1 *Complessità temporale*

Di seguito viene riportato l'algoritmo scritto in pseudocodice. I parametri sono D ed n, che sono rispettivamente la matrice delle distanze adattiva di dimensione $n \times n$ e la foglia n.

AlberoAdditivo(D, n**)**
se $n = 2$
  Restituisci l'albero costituito da un arco che collega le due foglie
altrimenti esegui le seguenti operazioni:
pesoArto $\leftarrow$ **limbweight**(n)
per $j \leftarrow 1$ **fino a** $n - 1$
  $D_{jn} \leftarrow D_{jn} - pesoArto$
  $D_{nj} \leftarrow D_{jn}$
(i,n,k) $\leftarrow$ tre foglie tale che $D_{ik} = D_{in} + D_{nk}$
$x \leftarrow D_{in}$
rimuovi la riga e la colonna n dalla matrice D
T $\leftarrow$**AlberoAdditivo(**D, $n - 1$**)**
$v \leftarrow$ il genitore di n che è distante x rispetto ad i in T, dove i è la foglia vicina di n
aggiungi la foglia n a T creando un arto tra v ed n il cui peso è *pesoArto*

Ogni volta che l'algoritmo richiama sé stesso, vengono eseguite le seguenti operazioni:

1. calcola il peso dell'arto dell'$n - esima$ foglia $\rightarrow T(step1) \simeq O(n^2)$[9];
2. aggiorna i valori della matrice $\rightarrow T(step2) \simeq O(n-1)$;
3. individua il punto in cui va inserita l'$n - esima$ foglia ed elimina la riga e la colonna n. Poiché questo step esegue sempre lo stesso numero di operazioni, possiamo considerare la sua complessità come costante $\rightarrow T(step3) \simeq O(1)$.

Quindi:

$$T(AlberoAdditivo) = T(step1) + T(step2) + T(step3) =$$

$$= O(n^2) + O(n-1) + O(1) \simeq O(n^2)$$

9 Si ricorda che la complessità di *limbweight* è $O(n^2)$.

Ma *AlberoAdditivo* viene eseguito n volte, ovvero fino a che non si costruisce un albero T con n foglie, pertanto la complessità totale dell'algoritmo è:

$$T(\mathtt{totale}) = T(\mathtt{AlberoAdditivo}) \times O(n) = O(n^2) \times O(n) = O(n^3)$$

4.5 ALGORITMO NEIGHBOR-JOINING

Gli algoritmi precedentemente elencati mostrano delle criticità, in quanto il primo riesce a costruire l'albero T solo se la matrice additiva D ha come elemento più piccolo una coppia di foglie vicine in T, mentre il secondo, anche se risolve questo problema, non riesce a costruirlo se D non è additiva.
È necessario fare una precisazione: non c'è modo che un albero si *adatti* ad una matrice *non additiva*, proprio per definizione della stessa, tuttavia è possibile costruire un albero che approssimi al meglio la distanza tra le foglie della matrice attraverso uno degli algoritmi più importanti della bioinformatica, il **Neighbor-Joining**. Nel caso in cui la matrice sia *additiva*, allora il **NJ** costruisce un albero che si adatta ad essa.
Si prenda in considerazione la seguente matrice *non additiva*[10]:

$$D = \begin{array}{c} \text{specie} \\ f \\ b \\ u \\ s \end{array} \begin{array}{c} \begin{array}{cccc} f & b & u & s \end{array} \\ \begin{pmatrix} 0 & 3 & 4 & 3 \\ 3 & 0 & 4 & 5 \\ 4 & 4 & 0 & 2 \\ 3 & 5 & 2 & 0 \end{pmatrix} \end{array}$$

L'obiettivo è quello di costruire un albero T che approssimi al meglio le distanze tra le foglie in D.
Il *primo* passo dell'algoritmo Neighbor-Joining consiste nel trovare D^*. Dato in input D si definisce D^* la seguente matrice [24]:

$$\forall f, b \in D,\ D^*(f,b) = (n-2) \cdot D(f,b) - \sum_{k=1}^{n} D(f,k) - \sum_{k=1}^{n} D(b,k)$$

dove $totalDistance(D_f) = \sum_{k=1}^{n} D(f,k)$ e $totalDistance(D_b) = \sum_{k=1}^{n} D(b,k)$. La sua caratteristica principale è che qualunque sia la matrice delle distanze in input, l'elemento più piccolo della relativa matrice D^* corrisponde ad una coppia di foglie *vicine* nell'albero T.
Quindi prima si trovano i valori di $totalDistance$ per ogni riga di D:

$$D = \begin{array}{c} \text{specie} \\ f \\ b \\ u \\ s \end{array} \begin{array}{c} \begin{array}{cccc} f & b & u & s \end{array} \\ \begin{pmatrix} 0 & 3 & 4 & 3 \\ 3 & 0 & 4 & 5 \\ 4 & 4 & 0 & 2 \\ 3 & 5 & 2 & 0 \end{pmatrix} \end{array} \quad \begin{array}{l} totalDistance(D_f) = 10 \\ totalDistance(D_b) = 12 \\ totalDistance(D_u) = 10 \\ totalDistance(D_s) = 10 \end{array}$$

10 Per brevità si definisce u=umano, s=scimpanzé, f=foca, b=balena.

Infine si calcolano i valori per creare $D^\star$:

- $D^\star(f,b) = (4-2)\cdot D(f,b) - \mathrm{totalDistance}(D_f) - \mathrm{totalDistance}(D_b) =$

  $= 2\cdot 3 - 10 - 12 = -16$

- $D^\star(f,u) = (4-2)\cdot D(f,u) - \mathrm{totalDistance}(D_f) - \mathrm{totalDistance}(D_u) =$

  $= 2\cdot 4 - 10 - 10 = -12$

- $D^\star(f,s) = (4-2)\cdot D(f,s) - \mathrm{totalDistance}(D_f) - \mathrm{totalDistance}(D_s) =$

  $= 2\cdot 3 - 10 - 10 = -14$

- $D^\star(b,u) = (4-2)\cdot D(b,u) - \mathrm{totalDistance}(D_b) - \mathrm{totalDistance}(D_u) =$

  $= 2\cdot 4 - 12 - 10 = -14$

- $D^\star(b,s) = (4-2)\cdot D(b,s) - \mathrm{totalDistance}(D_b) - \mathrm{totalDistance}(D_s) =$

  $= 2\cdot 5 - 12 - 10 = -12$

- $D^\star(u,s) = (4-2)\cdot D(u,s) - \mathrm{totalDistance}(D_u) - \mathrm{totalDistance}(D_s) =$

  $= 2\cdot 2 - 10 - 10 = -16$

Quindi la matrice è:

$$D^\star = \begin{array}{c} \text{specie} \\ f \\ b \\ u \\ s \end{array} \begin{array}{c} \begin{array}{cccc} f & b & u & s \end{array} \\ \begin{pmatrix} 0 & -16 & -12 & -14 \\ -16 & 0 & -14 & -12 \\ -12 & -14 & 0 & -16 \\ -14 & -12 & -16 & 0 \end{pmatrix} \end{array}$$

A questo punto si può passare al *secondo* step: si cerca l'elemento minimo in $D^\star$, ovvero $D^\star_{fb}$[11], le cui foglie f e b sono *vicine* in T.
Nel *terzo* invece si calcola il *delta* tra $\mathrm{totalDistance}(D_f)$ e $\mathrm{totalDistance}(D_b)$, quindi:

$$\Delta_{fb} = \frac{\mathrm{totalDistance}(D_f) - \mathrm{totalDistance}(D_b)}{n-2} = \frac{10-12}{4-2} = -1$$

11 Anche $D^\star_{us}$ è l'elemento più piccolo, tuttavia non se ne può scegliere più di uno alla volta.

Questo valore ci serve per il *quarto* step, ovvero trovare il peso dell'arto di f e di b:

$$limbweight(f) = \frac{D_{fb} + \Delta_{fb}}{2} = \frac{3 + (-1)}{2} = 1$$

$$limbweight(b) = \frac{D_{fb} - \Delta_{fb}}{2} = \frac{3 - (-1)}{2} = 2$$

Poiché si sono ottenute tutte le informazioni necessarie per poter inserire f e b nell'albero, si esegue lo step *cinque*: si inserisce il loro genitore non noto in D, ovvero una riga e colonna p tale che $\forall u \in D \setminus \{f, b\}$, $D_{up} = \frac{D_{fu}+D_{bu}-D_{fb}}{2}$ e si rimuovono le righe e le colonne di f e di b. Quindi:

$$D_{up} = \frac{D_{fu} + D_{bu} - D_{fb}}{2} = \frac{4 + 4 - 3}{2} = \frac{5}{2} = 2,5$$

$$D_{sp} = \frac{D_{fs} + D_{bs} - D_{fb}}{2} = \frac{3 + 5 - 3}{2} = \frac{5}{2} = 2,5$$

Si può notare che la distanza di entrambe le foglie dal genitore sia la stessa. La matrice risultante è:

$$D = \begin{array}{c} \text{specie} \\ p \\ u \\ s \end{array} \begin{array}{c} \begin{array}{ccc} p & u & s \end{array} \\ \begin{pmatrix} 0 & 2,5 & 2,5 \\ 2,5 & 0 & 2 \\ 2,5 & 2 & 0 \end{pmatrix} \end{array}$$

I cinque passi sopraelencati devono essere eseguiti fino a che non si ottiene una matrice 2×2, quindi, *step uno* $\rightarrow$ si calcolano $totalDistance(D_p)$, $totalDistance(D_u)$, $totalDistance(D_s)$ e si costruisce $D^\star$:

$$\begin{array}{l} totalDistance(D_p) = 5 \\ totalDistance(D_u) = 4,5 \\ totalDistance(D_s) = 4,5 \end{array} \quad D^\star = \begin{array}{c} \text{specie} \\ p \\ u \\ s \end{array} \begin{array}{c} \begin{array}{ccc} p & u & s \end{array} \\ \begin{pmatrix} 0 & -7 & -7 \\ -7 & 0 & -7 \\ -7 & -7 & 0 \end{pmatrix} \end{array}$$

Step due e tre $\rightarrow$ si sceglie un elemento minimo in $D^\star$ (ovvero $D^\star_{us}$) e si calcola $Delta_{us}$, quindi:

$$\Delta_{us} = \frac{totalDistance(D_u) - totalDistance(D_s)}{n - 2} = \frac{4,5 - 4,5}{3 - 2} = 0$$

Step quattro $\rightarrow$ si trovano i pesi degli arti di u e di s. Poiché il *delta* è uguale a zero, $limbweight(u)$ e $limbweight(s)$ hanno lo stesso valore:

$$limbweight(u) = \frac{D_{us} + \Delta_{us}}{2} = \frac{2 + 0}{2} = 1$$

$$limbweight(s) = \frac{D_{us} - \Delta_{us}}{2} = \frac{2-0}{2} = 1$$

Step cinque $\rightarrow$ si inserisce il genitore non noto k in D aggiornando i valori della matrice e si eliminano u ed s. Quindi:

$$D_{pk} = \frac{D_{up} + D_{sp} - D_{us}}{2} = \frac{2,5 + 2,5 - 2}{2} = \frac{3}{2} = 1,5$$

La matrice che si ottiene è la seguente:

$$D = \begin{array}{c} \\ p \\ k \end{array} \begin{array}{c} \begin{array}{cc} p & k \end{array} \\ \begin{pmatrix} 0 & 1,5 \\ 1,5 & 0 \end{pmatrix} \end{array}$$

Poiché D è di dimensione 2×2, si passa alla costruzione dell'albero. Dalla matrice si ricava che p e k sono dei nodi interni legati tramite un arco di peso 1,5, quindi:

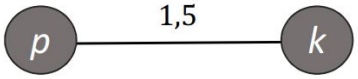

Figure 18: Albero iniziale T.

Come già specificato nei passi precedenti, p e k sono i genitori rispettivamente di f, b e u, s, i cui pesi degli arti sono già stati calcolati, pertanto si può costruire direttamente l'albero evolutivo finale T:

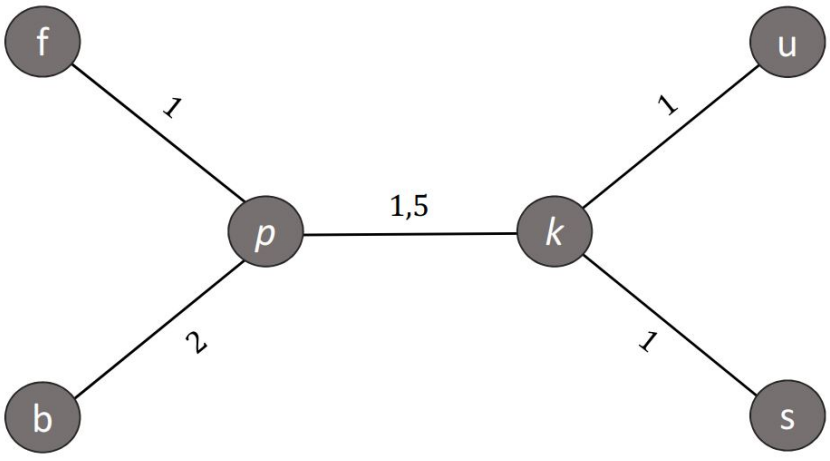

Figure 19: Albero T ottenuto dalla matrice delle distanze non additiva D.

Si può notare che l'albero ottenuto non si adatta completamente alla

matrice D in quanto, come detto all'inizio della sezione, non è possibile ottenerlo a causa della non additività di D. È comunque possibile misurare quanto T si adatti alla matrice attraverso la seguente formula [6]:

$$\mathrm{Discrepancy}(D(T), D) = \sum_{i=1}^{j-1} \sum_{j=i+1}^{n} (D_{ij}(T) - D_{ij})^2$$

Dove $D(T)$ è la matrice ottenuta da T, ovvero:

$$D(T) = \begin{array}{c} \text{specie} \\ f \\ b \\ u \\ s \end{array} \begin{array}{c} \begin{array}{cccc} f & b & u & s \end{array} \\ \begin{pmatrix} 0 & 3 & 3,5 & 3,5 \\ 3 & 0 & 4,5 & 4,5 \\ 3,5 & 4,5 & 0 & 2 \\ 3,5 & 4,5 & 2 & 0 \end{pmatrix} \end{array}$$

Pertanto $\mathrm{Discrepancy}(D(T), D)$ è:

$$\begin{aligned} &\mathrm{Discrepancy}(D(T), D) = (D_{1,2}(T) - D_{1,2})^2 + (D_{1,3}(T) - D_{1,3})^2 + \\ &+(D_{1,4}(T) - D_{1,4})^2 + (D_{2,3}(T) - D_{2,3})^2 + (D_{2,4}(T) - D_{2,4})^2 = \\ &= 0 + (3,5-4)^2 + (3,5-3)^2 + (4,5-4)^2 + (4,5-5)^2 + 0 = 1 \end{aligned}$$

Il risultato mostra che non c'è una grande discrepanza tra $D(T)$ e D, pertanto T approssima al meglio la matrice in input.

4.5.1 *Complessità temporale*

Si riporta l'algoritmo Neighbor-Joining in pseudocodice, dove D è la matrice delle distanze additiva o non e n è rispettivo il numero di righe e di colonne:

NeighborJoining(D, n**)**
se $n = 2$
  Restituisci l'albero T ottenuto da D costituito da un arco che collega le due foglie
altrimenti esegui le seguenti operazioni:
$D^\star \leftarrow$ matrice ottenuta da D
cerca l'elemento minimo $D^\star_{ij}$ in $D^\star$
$\Delta_{ij} \leftarrow \frac{totalDistance(D_i) - totalDistance(D_j)}{n-2}$
$limbweight(i) \leftarrow \frac{D_{ij}+\Delta_{ij}}{2}$
$limbweight(j) \leftarrow \frac{D_{ij}-\Delta_{ij}}{2}$
aggiungi una riga ed una colonna m a D tale che $\forall k \in D \setminus \{i, j\}$, $D_{km} = D_{mk} = \frac{D_{ki}+D_{kj}-D_{ij}}{2}$
elimina la righe e le colonne di i e di j da D
$T \leftarrow$ **NeighborJoining**(D, $n-1$)
connetti m alle foglie i e j nell'albero T
assegna all'arto di i il valore di $limbweight(i)$
assegna all'arto di j il valore di $limbweight(j)$
Restituisci T

Ogni volta che l'algoritmo viene richiamato, vengono eseguiti i seguenti step:

1. Crea $D^\star$ e cerca il suo elemento minimo. Supponendo di usare un algoritmo di ricerca lineare, il cui costo nel caso pessimo è $O(n)$ (per un vettore di lunghezza n), nel caso di una matrice di dimensione $n \times n$ si deve eseguire questa operazione per tutte le n righe, quindi:

$$T(step1) = O(n) \times O(n) = O(n^2)$$

2. Calcola il Delta ed i pesi degli arti di i e di j. A differenza di quanto visto nell'algoritmo "*Albero Additivo*", la complessità di *limbweight* non è $O(n^2)$ ma $O(1)$. Questo perché in questo caso si conosce

subito quale è la foglia e quindi si può applicare direttamente la formula per calcolare il peso del suo arto. Quindi:

$$T(step2) = O(1)$$

3. Aggiunge il genitore di i e di j e calcola la sua distanza rispetto agli altri elementi:

$$T(step3) = O(n-2) \simeq O(n)$$

La complessità di *NeighborJoining* è:

$$T(NeighborJoining) = T(step1) + T(step2) + T(step3) =$$

$$= O(n^2) + O(1) + O(n) \simeq O(n^2)$$

NeighborJoining viene eseguito n volte, quindi la complessità totale è:

$$T(Totale) = T(NeighborJoining) \times O(n) = O(n^2) \times O(n) = O(n^3)$$

4.6 UNWEIGHTED PAIR GROUP METHOD WITH ARITHMETIC MEAN

Nelle sezioni precedenti sono stati mostrati degli algoritmi che risolvono il *problema degli alberi basati sulla distanza* costruendo alberi evolutivi *senza radice*. In particolare il Neighbor-Joining si è rivelato il più efficace poiché risolve tutte le criticità relative agli algoritmi precedenti.
UPGMA (Unweighted Pair Group Method with Arithmetic Mean) è un algoritmo che, data in input una matrice delle distanze D additiva o non, restituisce un albero *radicato* T in cui tutte le foglie sono alla stessa distanza dalla radice (albero ultrametrico) [6].
Si riporta di seguito un esempio:

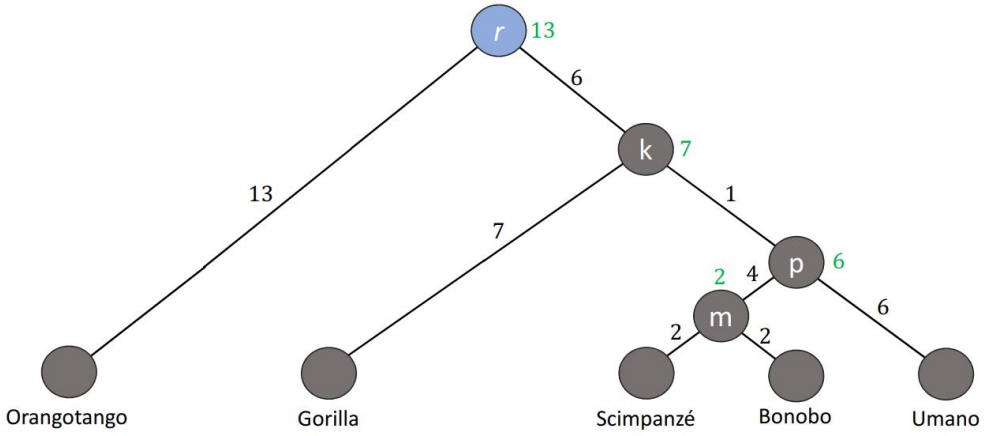

Figure 20: Esempio di albero ultrametrico.

In tale albero i nodi interni rappresentano le *speciazioni*, quei processi attraverso i quali si formano nuove specie. Ogni volta che si verifica una speciazione una linea evolutiva viene divisa in due [25] e quindi da un antenato deriveranno due discendenti. Ad esempio nella figura 20 si può notare che in un certa epoca la specie m si è suddivisa in due specie diverse (a causa di mutazioni), ovvero scimpanzé e bonobo. I numeri in verde accanto ai nodi indicano l'età (in milioni di anni), mentre il peso degli archi è dato dalla differenza tra le età dei nodi, ad esempio il peso dell'arco che collega k e p è dato dalla differenza delle loro età ($7 - 6 = 1$) e così via per tutti gli altri archi. Poiché le foglie dell'albero rappresentano le specie ancora non suddivise, la loro età è uguale a zero.

L'algoritmo *UPGMA* può essere diviso in più fasi. Si prenda in considerazione la seguente matrice *non additiva*[12]:

$$D = \begin{array}{c} \text{specie} \\ f \\ b \\ u \\ s \end{array} \begin{array}{c} \begin{array}{cccc} f & b & u & s \end{array} \\ \begin{pmatrix} 0 & 3 & 4 & 3 \\ 3 & 0 & 4 & 5 \\ 4 & 4 & 0 & 2 \\ 3 & 5 & 2 & 0 \end{pmatrix} \end{array}$$

Si può notare che è la stessa matrice usata nella sezione dedicata al Neighbor-Joining.
Fase uno $\rightarrow$ a partire da D si formano n cluster[13], ciascuno contenente una foglia, dove n è il numero di righe (o colonne) della matrice.

Figure 21: Cluster iniziale.

Fase due $\rightarrow$ si scelgono i due cluster X e Y più vicini secondo la seguente definizione di distanza [6]:

$$D_{X,Y} = \frac{1}{|X| \cdot |Y|} \cdot \sum_{i \in X,\, j \in Y} D_{i,j}$$

dove i rappresentano gli elementi del cluster X, j del cluster Y, $D_{i,j}$ è la distanza tra i e j, $|X|$ e $|Y|$ sono rispettivamente il numero degli elementi in X ed in Y. Poiché in questo caso ciascun cluster è formato da un solo elemento, prendere i due cluster più vicini equivale a scegliere l'elemento più piccolo in D, ovvero $D_{u,s} = 2$.
Fase tre $\rightarrow$ si crea un cluster Z che è dato dall'unione di X ed Y. Quindi, nel caso in esame, si crea un cluster $\{u, s\}$ tale che $\{u, s\} = u \cup s$. Il risultato di questo passo è il seguente:

12 Per brevità si definisce u=umano, s=scimpanzé, f=foca, b=balena.
13 Insiemi di elementi che condividono una determinata proprietà [13].

Figure 22: Il cluster $\{u, s\}$ è stato ottenuto attraverso l'unione di u ed s.

Fase quattro $\rightarrow$ si forma un nodo interno per Z la cui età è $age(Z) = \frac{D_{X,Y}}{2}$ e si calcola il peso degli archi[14]. Applicato all'esempio si ottiene che:

$$age(\{u, s\}) = \frac{D_{u,s}}{2} = \frac{2}{2} = 1$$

$$edgeweight(\{u, s\}, s) = age(\{u, s\}) - age(s) = 1 - 0 = 0$$

$$edgeweight(\{u, s\}, u) = age(\{u, s\}) - age(u) = 1 - 0 = 0$$

L'albero risultante è:

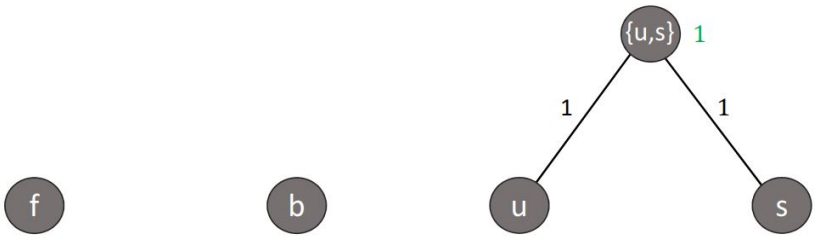

Figure 23: Albero T dopo aver calcolato l'età del nodo {u, s} ed il peso degli archi.

Fase cinque $\rightarrow$ Si aggiorna la matrice D calcolando la distanza tra i cluster usando la formula $D_{X,Y} = \frac{1}{|X| \cdot |Y|} \cdot \sum_{i \in X,\, j \in Y} D_{i,j}$, che è quella usata nella fase due. Quindi:

$$D = \begin{array}{c} \text{specie} \\ f \\ b \\ \{u, s\} \end{array} \begin{array}{c} \begin{array}{ccc} f & b & \{u, s\} \end{array} \\ \begin{pmatrix} 0 & 3 & 3,5 \\ 3 & 0 & 4,5 \\ 3,5 & 4,5 & 0 \end{pmatrix} \end{array}$$

Dove i nuovi valori sono ottenuti da:

$$D_{f,\{u,s\}} = \frac{D_{f,u} + D_{f,s}}{2} = \frac{4+3}{2} = \frac{7}{2} = 3,5$$

14 Come detto poc'anzi, il peso dell'arco che collega Z ad X è uguale alla differenza tra le loro età.

$$D_{b,\{u,s\}} = \frac{D_{b,u} + D_{b,s}}{2} = \frac{4+5}{2} = \frac{9}{2} = 4,5$$

A questo punto vanno eseguite le cinque fasi sopraelencate fino a che non otteniamo un unico cluster contenente tutte le righe (o colonne) di D.

Fase uno, due e tre → si formano n cluster a partire dalla nuova matrice D, si sceglie l'elemento più piccolo ($D_{f,b}$) e si crea un cluster $\{f, b\}$ a partire da f e da b, quindi:

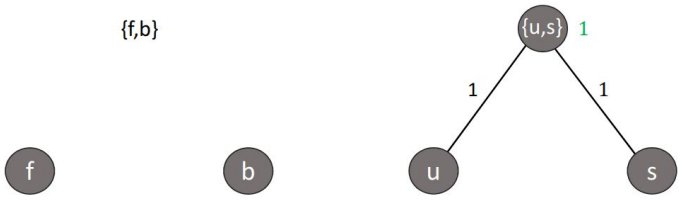

Figure 24: Il cluster $\{f, b\}$ è stato ottenuto attraverso l'unione di f ed b.

Fase quattro → si forma un nodo interno per il cluster $\{f, b\}$ e si calcola sia la sua età che il peso dei suoi archi incidenti, quindi:

$$age(\{f, b\}) = \frac{D_{f,b}}{2} = \frac{3}{2} = 1,5$$

$$edgeweight(\{f, b\}, b) = age(\{f, b\}) - age(b) = 1,5 - 0 = 1,5$$

$$edgeweight(\{f, b\}, f) = age(\{f, b\}) - age(f) = 1,5 - 0 = 1,5$$

L'albero risultante è:

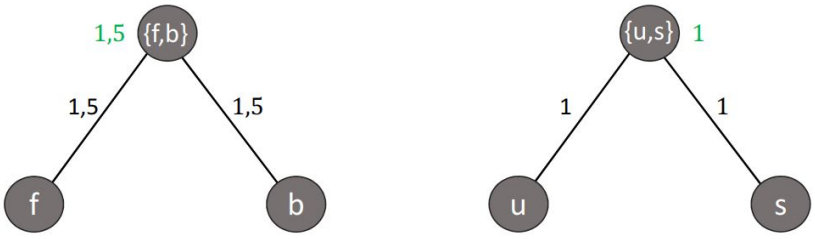

Figure 25: Albero T dopo aver calcolato l'età del nodo $\{f, b\}$ ed il peso degli archi.

Fase cinque → Si aggiorna la matrice D calcolando la distanza tra i cluster usando la formula $D_{X,Y} = \frac{1}{|X|\cdot|Y|} \cdot \sum_{i \in X, j \in Y} D_{i,j}$. La matrice risultante D è:

$$D = \begin{array}{c} \text{specie} \\ \{f, b\} \\ \{u, s\} \end{array} \begin{array}{c} \begin{array}{cc} \{f, b\} & \{u, s\} \end{array} \\ \begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix} \end{array}$$

dove l'elemento $D_{\{f,b\},\{u,s\}}$ è ottenuto da:

$$D_{\{f,b\},\{u,s\}} = \frac{D_{f,\{u,s\}} + D_{b,\{u,s\}}}{2} = \frac{3,5+4,5}{2} = \frac{8}{2} = 4$$

Poiché ancora non si è ottenuto un unico cluster contenente tutte le righe (o colonne) di D, bisogna eseguire nuovamente gli step. Quindi:

- l'elemento più piccolo è $D_{\{f,b\},\{u,s\}}$;
- si crea un nuovo cluster $\{f, b, u, s\}$ e si forma un nodo interno per esso, che sarà la radice dell'albero T. L'età è $age(\{f, b, u, s\}) = \frac{D_{\{f,b\},\{u,s\}}}{2} = \frac{4}{2} = 2$, mentre gli archi sono $age(\{f, b, u, s\}) - age(\{f, b\}) = 2 - 1,5 = 0,5$ e $age(\{f, b, u, s\}) - age(\{u, s\}) = 2 - 1 = 1$.
  L'albero finale T è:

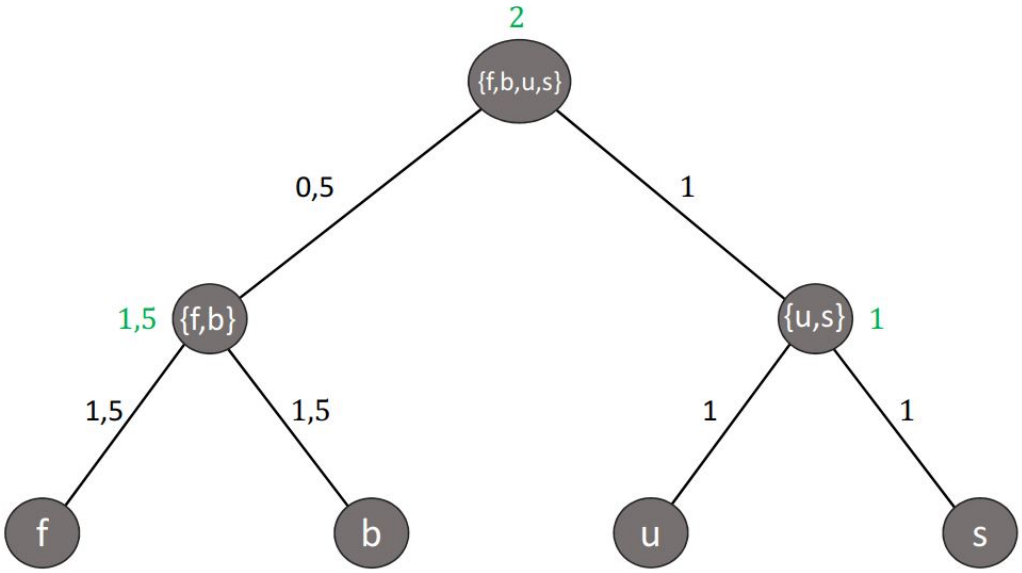

Figure 26: Albero finale con radice T ottenuto dalla matrice D.

- si aggiorna la matrice D, ottenendo:

$$\begin{array}{c} \text{specie} \quad \{f, b, u, s\} \\ D = \{f, b, u, s\} \begin{pmatrix} 2 \end{pmatrix} \end{array}$$

  dove $D_{\{f,b,u,s\}} = \frac{D_{\{f,b\},\{u,s\}}}{2} = \frac{4}{2} = 2$.

Poiché abbiamo ottenuto l'albero T e la matrice D è composta da un unico cluster, l'algoritmo è terminato.

4.6.1 *Complessità temporale*

Sia D la matrice delle distanze in input[15] e n il numero di righe o di colonne in D. L'algoritmo UPGMA scritto in pseudocodice è il seguente:

UPGMA(D, n**)**
Clusters $\leftarrow$ n cluster, ciascuno composto da un elemento di D
costruisci un grafo T composto da n vertici isolati. Questi rappresentano i cluster
$age(v) \leftarrow 0 \ \forall$ node $v \in T$
fino a che c'è più di un cluster in Clusters
  cerca i cluster più vicini X e Y
  unisci X ed Y in un cluster Z composto da $X + Y$ elementi
  aggiungi un nodo interno Z in T
  connetti Z ad X e Y tramite due archi
  calcola l'età di Z
  rimuovi la riga e la colonna di X e di Y dalla matrice D
  rimuovi i cluster X e Y dall'insieme Clusters
  aggiorna la matrice D, quindi aggiungi una riga ed una colonna Z e calcola la sua distanza rispetto a tutti gli altri cluster in Clusters
  aggiungi Z all'insieme Clusters
radice $\leftarrow$ il nodo nell'albero T corrispondente all'ultimo cluster rimasto
per ogni arco (v, w) in T
  $edgeweight(v, w) \leftarrow age(v) - age(w)$
restituisci T

Si evidenzia che:

- Nella fase iniziale viene creato un insieme di n cluster (Clusters) e l'età di tutti i vertici in T viene impostata uguale a zero. Poiché quest'ultima operazione viene fatta n volte $\rightarrow T(step1) = O(n)$.
- Vengono iterate una serie di azioni sui cluster calcolati in precedenza e poiché in totale sono n, si ha che $\rightarrow O(n)$. L'insieme di queste operazioni ha una complessità di $O(n)$, in quanto ad ogni iterazione è necessario aggiornare la matrice D ricalcolando la distanza tra i suoi elementi, che sono n. In altre parole, vengono eseguite n azioni n volte. Questo significa che $\rightarrow T(step2) = O(n) \times O(n) = O(n^2)$.

15 Si ricorda che la matrice può essere additiva o non additiva.

- Prima di restituire l'albero T viene calcolato il peso di tutti gli archi, che sono $n-1$, quindi $\rightarrow T(step3) = O(n-1) \simeq O(n)$.

Si può concludere che la complessità totale dell'UPGMA è:

$$T(totale) = T(step1) + T(step2) + T(step3) = O(n) + O(n^2) + O(n) \simeq O(n^2)$$

BIBLIOGRAPHY

[1] Allegretti M. (2014).
La biologia strutturale in movimento.
AIRInforma: AIRIcerca.
http://informa.airicerca.org/it/2014/10/04/biologia-strutturale-in-movimento/

[2] Bacic, A., Likić, V. A., Lithgow, T., McConville, M. J.(2010).
Systems Biology: The Next Frontier for Bioinformatics.
Advances in Bioinformatics, 2010, 1–10.
DOI:
l10.1155/2010/268925

[3] Bateman, A., Peng, H., Valencia, A., Wren, J. D. (2012).
Bioimage informatics: a new category in Bioinformatics.
Bioinformatics, 28(8), 1057–1057.
DOI:
10.1093/bioinformatics/bts111

[4] Bayat Ardeshir (2002).
Science, medicine, and the future: Bioinformatics.
BMJ, 324(7344), 1018–1022. DOI:
10.1136/bmj.324.7344.1018

[5] Basic Local Alignment Search Tool Website.
BLAST.
https://blast.ncbi.nlm.nih.gov/Blast.cgi (Cited on page 15.)

[6] Baum O. J, Zveibil M.
Understanding Bioinformatics.
New York And London, Garland Science, 2007. (Cited on pages 53, 56, and 57.)

[7] Bear R., Herren C., Horne E., Rintoul D., Smith-Caldas M., Snyder B.
Building a phylogenetic tree.
https://www.khanacademy.org/science/biology/her/tree-of-life/a/building-an-evolutionary-tree (Cited on page 25.)

[8] Berg L., Martin W. D., Solomon E.
Biology.
Brooks Cole, 2010. (Cited on page 10.)

[9] Berk A., Darnell J., Kaiser A. C., Krieger M., Lodish H., Matsudaira P., Scott P. M., Zipursky L., .
Molecular Cell Biology.
W. H. Freeman (2008). (Cited on page 10.)

[10] Borne, K. D.
X-Informatics: Practical Semantic Science.
`http://adsabs.harvard.edu/abs/2009AGUFMIN43E..01B`.
George Mason University Fairfax VA, American Geophysical Union Fall Meeting, 12/2009.

[11] Breitling, R. (2010).
What is systems biology? Frontiers in Physiology.
DOI:
`10.3389/fphys.2010.00009`

[12] Cain M., Minorsky P., Reece J, Urry L., Wasserman S.
Campbell biology.
Pearson Higher Education, 2016. (Cited on pages 9 and 10.)

[13] Cambridge Dictionary.
Cluster.
`https://dictionary.cambridge.org/dictionary/english/cluster` (Cited on page 57.)

[14] Candavelou, M., Gollapalli, S., Gopal,J., Karthikeyan, K., Venkatesan, A.(2013).
Computational Approach for Protein Structure Prediction.
Healthcare Informatics Research, 19(2), 137.
DOI:
`10.4258/hir.2013.19.2.137`

[15] Can T. (2013).
Introduction to Bioinformatics.
miRNomics: MicroRNA Biology and Computational Analysis.
DOI:
`10.1007/978-1-62703-748-8_4` (Cited on page 13.)

[16] Charette, S. J., Derome, N., Gauthier, J., Vincent, A. T.(2018).
A brief history of bioinformatics.
Briefings in Bioinformatics.
DOI:
10.1093/bib/bby063

[17] Chen J., Sidhu S. A.
Biological Database Modeling.
Norwood, MA, Artech House, 2008.

[18] Choudhuri S.
Bioinformatics For Beginners.
Maryland, Elsevier Inc, 2014. (Cited on page 25.)

[19] Compeau P., Pevzner P.
Bioinformatics Algorithms, An Active Learning Approach, Vol II.
United States Of America, Active Learning Publishers, 2015. (Cited on pages 26, 27, 31, and 40.)

[20] Dennis E. Shasha, Hannu Toivonen, Jason T. L. Wang, Mohammed J. Zaki.
Data Mining in Bioinformatics.
London, Springer, 2004. (Cited on pages 14, 21, 29, and 40.)

[21] Durbin R., Eddy S., Krogh A., Mitchison G.
Biological Sequence Analysis. Probabilistic models of proteins and nucleic acids.
Cambridge University, Cambridge University Press, 1998.

[22] Emery L.
Phylogenetics: An introduction.
European Bioinformatics Institute-European Molecular Biology Laboratory.
https://www.ebi.ac.uk/training/online/course/introduction-phylogenetics
Oxford.

[23] Ensembl Website.
Ensembl.
http://www.ensembl.org/ (Cited on page 23.)

[24] Gascuel O., Steel M (2006).
Neighbor-Joining Revealed.

LIRMM, Montpellier, France and Biomathematics Research Centre, University of Canterbury, Christchurch, New Zealand.
DOI:
10.1093/molbev/msl072 (Cited on page 49.)

[25] Gittleman L. J.
Speciation.
Enciclopedia Britannica.
https://www.britannica.com/science/speciation (Cited on page 56.)

[26] Haque Sultan Omar (2016).
Phylogenetics.
Enciclopedia Britannica.
https://www.britannica.com/science/phylogenetics#accordion-article-history

[27] ICAR CNR: Istituto Di Calcolo E Reti Ad Alte Prestazioni.
Bioinformatica.
https://www.icar.cnr.it/bio-informatica/ (Cited on page 13.)

[28] Jiang, X., Lin, G., Liu, X., Zeng, X., Zou, Q.(2018).
Sequence clustering in bioinformatics: an empirical study.
Briefings in Bioinformatics.
DOI:
https://doi.org/10.1093/bib/bby090

[29] Jones C. Neil and Pevzner A. Pavel.
An introduction to bioinformatics algorithms.
Massachusetts, Massachusetts Institute of Technology, 2004. (Cited on pages 30 and 40.)

[30] MathWorks.
What Is the Genetic Algorithm?
https://it.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

[31] Mallawaarachchi V (2017).
Introduction To Genetic Algorithms-Including Example Code.
https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[32] Mount W. David.
Bioinformatics: Sequence and Genome Analysis.
Cold Spring Harbor Laboratory Press, 2004.

[33] National Human Genome Research Institute (2015).
DNA Sequencing Fact Sheet.
`https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Fact-Sheet`

[34] National Center for Biotechnology Information Website.
NCBI.
`https://www.ncbi.nlm.nih.gov/` (Cited on page 23.)

[35] Olivieri M., Verri M. C.
Algoritmi E Strutture Dati II.
Anno Accademico 2013/2014. (Cited on pages 25 and 26.)

[36] Proteine Data Bank Website.
PDB.
`http://www.rcsb.org/` (Cited on pages 19, 22, and 23.)

[37] Raut, A., Raut, S. A., Sathe, S. R.(2010).
Bioinformatics: Trends in gene expression analysis.
International Conference on Bioinformatics and Biomedical Technology.
DOI:
`10.1109/icbbt.2010.5479003`

[38] Systems Biology Workbench Website.
SBW.
`http://sbw.sourceforge.net/` (Cited on page 21.)

[39] Semple C., Steel M.
Phylogenetics.
Oxford, Oxford University Press, 2003.

[40] Systems Biology Markup Language.
SBML.
`http://sbml.org/` (Cited on page 21.)

[41] Templeton R. Alan.
Population Genetics and Microevolutionary Theory.
Washington University, St. Louis, Missouri, Wiley-Liss, 2006.

[42] Tramontano Anna (2003).
La grande scienza. Bioinformatica.
http://www.treccani.it/enciclopedia/la-grande-scienza-bioinformatica_%28Storia-della-Scienza%29/ (Cited on page 13.)