Università degli Studi di Pisa

Facoltà di Scienze Matematiche Fisiche e Naturali
Dipartimento di Informatica

Laurea Magistrale in INFORMATICA

Algoritmi efficienti per la scoperta di pattern ripetuti a intervalli

Rosario Lombardo

Controrelatrice
Prof. Maria Grazia Scutellà

Relatori
Prof. Roberto Grossi
Prof. Roberto Marangoni

18 Luglio 2008
Anno Accademico 2007–2008

A mamma, papà e Franzuà.

Ai 118!

Si comincia.

Indice

Ringraziamenti

Le persone che hanno contribuito, per semplice cortesia o involontariamente, al completamento del lavoro sono molte di più di quelle che desidero citare qui; queste ultime, dunque, meritano un riconoscimento particolare.

Non avendo sufficiente tempo per la mansione, riporto invece un programma *equivalente*.

Algoritmo 0.1 Ringraziamenti

```
1: extern Hashtable people;
2: people.Keys.SortAscending();
3: for all Person p in people do
4:    print p.Name.Format() + ‘‘ grazie per ’’ + p.thanks;
5: end for
```

L'output dell'algoritmo è il seguente:

Daniela grazie per aver riletto più volte tutta la tesi;

Prof. Grossi grazie per le innumerevoli indicazioni necessarie al completamento dei contenuti nei tempi prestabiliti;

Prof. Marangoni grazie per l'ampia disponibilità dimostratami anche a distanza;

Papà grazie per essere stato sempre presente con il tuo costante incoraggiamento di una vita.

Rosario

Introduzione

Lo studio del genoma fornisce la base non solo per la conoscenza delle caratteristiche della nostra specie, ma anche per comprendere tutta la serie di modifiche intercorse tra specie e specie e tra individui della stessa specie ma che mostrano tratti differenti. Le ricadute del Progetto Genoma non sono limitate al settore della salute umana, ma giungono ad interessare campi di varia natura. Come termine ultimo, lo studio del genoma ha delle notevoli applicazioni in campo biotecnologico, le quali possono rivestire grande importanza economica.

Nel genoma sequenziato è stata rilevata, oltre ai circa 30.000 geni, un'enorme quantità di sequenze di cui ancora non si conoscono funzionamento e scopo. A seconda delle possibili considerazioni, la maggior parte dell'intera sequenza viene reputata non codificante, o "*DNA-spazzatura*", in una stima che varia dal 72% al 98%. Sono dunque molti gli sforzi che si concentrano sull'analisi della sequenza di nucleotidi e con questa tesi si vuol proporre un nuovo contributo in questo contesto.

L'analisi formale delle sequenza ha sviluppato il concetto di *cluster* di geni, ovvero delle regioni "interessanti" dal punto di vista dell'analisi biologica. I cluster possono essere visti come stringhe con ripetizioni oppure permutazioni di elementi tutti distinti. Entrambi gli approcci sono stati variamente sviluppati in letteratura e presentano peculiarità specifiche.

Utilizzando il modello a permutazioni, è stato fatto ricorso alla proprietà esibita dagli Alberi PQ di catturare un particolare tipo di pattern in una coppia di permutazioni. Infatti i nodi dell'albero PQ mettono in evidenza i segmenti

contigui da quelli variamente permutati. La proprietà si applica a permutazioni con elementi unici, oppure a sequenze di geni con pochissime ripetizioni opportunamente preprocessate. Di recente, si è però tentata l'applicazione anche a sequenze nucleotidiche altamente ridondanti.

Il fondamentale problema di codificare due stringhe ridondante in termini di permutazioni resta un problema aperto, a cui si è risposto essenzialmente in due modi: 1) eliminando i duplicati secondo un qualche criterio, oppure 2) assegnando un'enumerazione fra quelle possibili che, nel caso di sequenze nucleotidiche, sono in numero fattoriale.

Dal momento che non esiste ancora sufficiente conoscenza per costruire un adeguato modello dell'informzione per filtrare sequenze di caratteri rilevanti da altre apparentemente senza significato, si ricorre ad approcci che non prevedono l'uso di un modello di dati. La formulazione di notazioni massimali, senza introdurre perdita d'informazione, cattura importanti caratteristiche sulla struttura interna dei motif e ne riduce drasticamente il numero da analizzare, conferendo un senso all'enorme numero di risultati.

In questa tesi propongo un approccio alla codifica delle sequenze nucleoditiche altamente ridondanti, tale da produrre permutazioni numeriche. Questa metodologia, che usa i Suffix Tree, mira a generare codifiche esenti da alterazioni o artefatti dell'informazione genetica producendo, per design, permutazioni "compatibili" con i *minimal consensus PQ Tree*, i quali sono usati per la creazione della notazione massimale. Si forniscono le motivazioni che hanno ispirato questo approccio, basato sull'analisi della struttura permutativa interna delle stringhe. Si delineano alcuni possibili sviluppi e si forniscono molti esempi accompagnati da un caso d'uso reale.

Questa tesi è consultabile online all'indirizzo `http://etd.adm.unipi.it/theses/available/etd-06182008-085952`.

Organizzazione della tesi

La tesi di laurea si sviluppa procedendo dai concetti generali per specializzarsi sempre di piu sull'aspetto informatico applicato alla genetica.

- Nel capitolo 1 si fa una panoramica biologica che chiarisce l'intrigata bellezza del nostro patrimonio genetico, mettendo in evidenza le richieste poste agli informatici;
- si continua con lo stato dell'arte in ambito accademico (§ 2.1–2.5) seguito dallo stato dell'arte nell'ambiente operativo dei genetisti (§ 2.6);
- nel capitolo 3 si procede esponendo le motivazioni comuni ai due approcci e gli obiettivi che hanno guidato il lavoro di tesi (§ 3.1);
- si presentano i risultati di un primo approccio basato sugli intervalli comuni con ripetizioni (§ 3.2) traendone le dovute considerazioni;
- dopo le motivazioni fornite al cap. 3, nel cap. 4 si approfondisce la mia proposta metodologica per la codifica delle sequenze biologiche altamente ridondanti.
- si conclude indicando i possibili sviluppi e alcuni problemi aperti;
- in Appendice A si fornisce un caso d'uso interattivo realmente applicato nel campo della bioinformatica.

Capitolo 1

Nozioni biologiche

In questo capitolo[1] si vuole introdurre il lettore, anche attraverso illustrazioni[2] esemplificative, alle conoscenze di genetica che ispirano gli approcci algoritmici usati nell'analisi di sequenze genomiche. In particolare, l'attenzione sarà focalizzata su quegli approcci di tipo "*discovery*", ovvero orientati all'identificazione di sequenze reputate interessanti secondo determinati criteri assegnati a priori.

Il capitolo si sviluppa partendo dalle basi biologiche, per procedere gradualmente con la descrizione delle nozioni necessarie a comprendere la natura del DNA e, più in particolare, per chiarire due aspetti fondamentali:

1. l'intimo legame tra la sequenza di *nucleoditi* e la struttura del DNA;
2. la caratterizzazione strutturale di segmenti funzionalmente diversi della sequenza.

Si intraprende lo studio del DNA sotto forma di sequenza con l'intenzione di riuscire a dedurne informazioni sulla sua struttura interna e, dunque, sulla sua possibile funzione biologica.

[1] Basato sul materiale, gentilmente rilasciato in pubblico dominio, del Prof. Roberto Marangoni, "Basi biologiche dell'approccio *discovery* nell'analisi testuale di sequenze genomiche". Occorre precisare che si è spesso ricorso a molte semplificazioni per mantenere in termini concisi e introduttivi nozioni e concetti estremamente vasti. Questo materiale, quindi, non ha il livello adeguato ad una trattazione specialistica di area biologica.

[2] Le figure, qualora non siano originali, sono tratte da Internet e, almeno nelle fonti originarie, non recano indicazioni di copyright, eccetto ove diversamente specificato.

1.1 Il DNA

L'acido deossiribonucleico, o DNA, è un polimero organico costituito da quattro monomeri[3] detti nucleoditi. I nucleotidi sono caratterizzati da una *base azotata* alla quale si legano lo zucchero deossiribosio ed un gruppo fosfato. Le quattro basi azotate che intervengono nella formazione dei nucleotidi del DNA sono:

- Adenina (A)
- Citosina (C)
- Guanina (G)
- Timina (T)

Nonostante i termini *base* e *nucleotide* indichino evidentemente composti chimici diversi, si parla spesso di basi *in vece* dei nucleotidi, dal momento che è sempre la base a qualificare chimicamente l'intero gruppo, mentre lo zucchero e il fosfato rimangono componenti costanti.

Ogni nucleotide è costituito da uno scheletro laterale (zucchero e gruppo fosfato), che ne permette il legame covalente[4] con i nucleotidi adiacenti, formando così il singolo filamento, mentre la base azotata instaura legami idrogeno con la corrispondente base azotata presente sul filamento opposto.

L'appaiamento fra le basi è reso possibile dalla complementarità chimico-fisica che permette alla A di appaiarsi con la T, e alla C di appaiarsi con la G per mezzo dei legami idrogeno (si veda la Figura 1.1): la coppia A–T stabilisce due legami idrogeno, mentre la coppia C–G ne stabilisce tre.

Il legame idrogeno presente fra le basi è molto più debole dei legami covalenti presenti sul filamento, e questo spiega perché serva molta più energia per spezzare

[3] Un *polimero* (dal greco *molte parti*) è una macromolecola costituita da numerose molecole più piccole, i monomeri. I *monomeri* sono dotati di gruppi funzionali che li mettono in grado di combinarsi ricorsivamente con altre molecole – identiche a sé o *reattivamente* complementari a sé – a formare macromolecole più lunghe.

[4] Legame chimico di tipo forte, in quanto mantiene vicini i nuclei atomici condividendo gli elettroni.

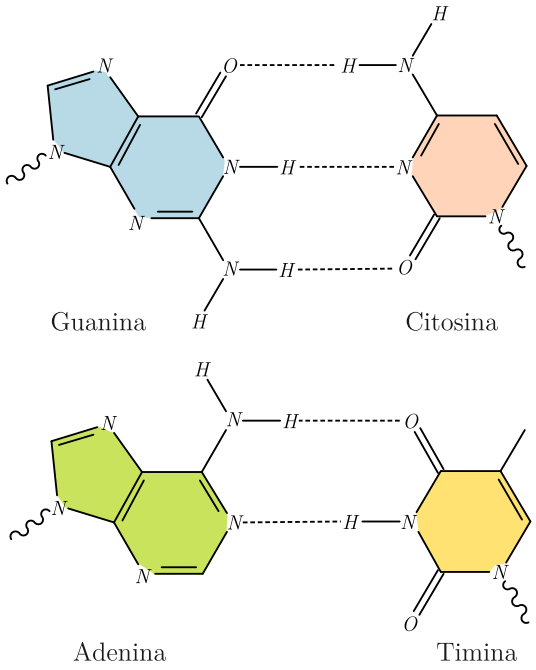


Figura 1.1: In alto, un appaiamento G–C, caratterizzato da tre legami idrogeno. In basso, un appaiamento A–T con due legami idrogeno. I legami idrogeno sono segnati con linea tratteggiata ad evidenziare la minore resistenza.

un filamento e, al contrario, sia relativamente facile separare fra loro la coppia di filamenti.

Le interazioni descritte producono filamenti di notevole lunghezza. Ad esempio, il più grande cromosoma umano (il cromosoma 1) contiene quasi 250 milioni di paia di basi. Il numero di basi appaiate disposte sul doppio filamento viene comunemente usato come misura della lunghezza di una molecola di DNA; questa unità di misura pratica si indica con *bp* (dall'inglese *base pair*).

Ogni filamento ha due estremità chiamate *estremità 5'* ed *estremità 3'*. Negli organismi viventi, il DNA non è quasi mai presente sotto forma di singolo filamento, ma piuttosto come una molecola costituita da due filamenti appaiati e antiparalleli.

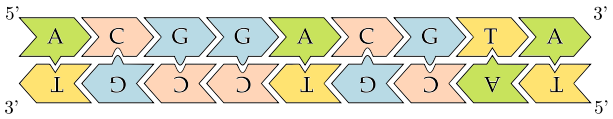


Figura 1.2: I due filamenti appaiati e antiparralleli della molecola di DNA.

Nucleotidi consecutivi sono collegati unendo l'estremità 3' dell'uno all'estremità 5' dell'altro come mostrato in Figura 1.2. Le estremità 5' e 3' dei due filamenti vanno in direzioni opposte definendo quindi un filamento *senso* a cui se ne appaia un altro con orientazione opposta detto *antisenso*. I filamenti si intrecciano tra loro a formare una struttura a *doppia elica*.

Da una prospettiva di elaborazione delle informazioni, la conoscenza di uno solo dei due filamenti è sufficiente per risalire all'intera molecola. Tuttavia l'informazione contenuta su un filamento è decodificata sequenzialmente andando dall'estremità 5' verso l'estremità 3', e ciò permette ad entrambi i filamenti di trasportare informazione biologica rilevante. L'importanza di questo aspetto diverrà evidente fra poco, quando verrà illustrata la duttilità del DNA, le molteplici conformazioni che può assumere e le proprietà di interazione col macchinario molecolare della cellula che sono funzione della topologia locale di specifici tratti del DNA.

1.2 Struttura del DNA

Il DNA rappresenta il materiale genetico responsabile dell'informazione ereditaria della maggior parte degli organismi: ad eccezione di virus con solo RNA, tutti gli altri esseri viventi usano, per veicolare le informazioni ereditarie, il DNA.

Una coppia complementare di filamenti di DNA, in soluzione fisiologica, assume la struttura a doppia elica del celebre modello di Watson e Crick. La doppia elica è una spirale destrorsa che prende forma dall'avvitamento su se stessi dei due filamenti, lasciando esposti dei solchi a due larghezze. La differente ampiezza dei

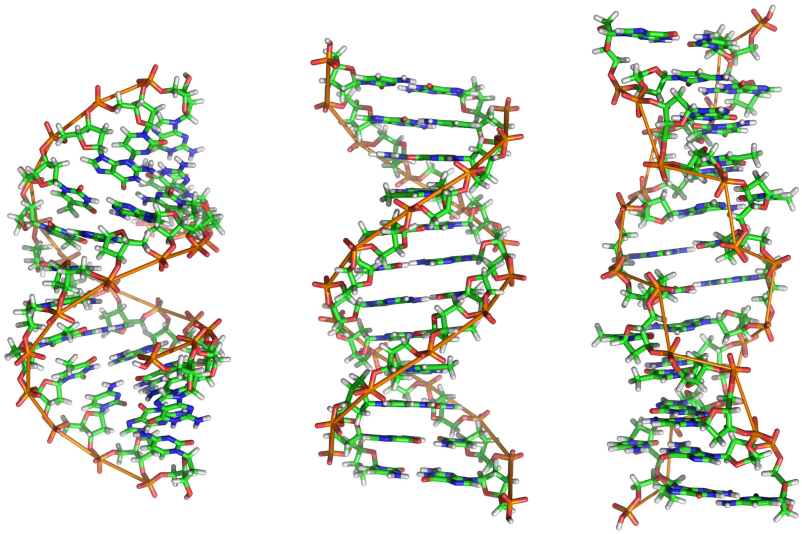


Figura 1.3: Da sinistra a destra le tre conformazioni di DNA presenti in natura: A-DNA, B-DNA e Z-DNA.

due solchi si traduce concretamente in una differente accessibilità alle basi azotate, a seconda che si trovino nel solco maggiore o minore. Molte proteine, quali per esempio i fattori di trascrizione[5], prendono quindi contatto principalmente con le basi presenti nel solco maggiore.

Il modello a doppia elica di Watson e Crick risulta una delle possibili strutture che il DNA può assumere. All'interno dello schema generale dell'antiparallelismo e della complementarità dei filamenti, il DNA può variare in modo sorprendente la propria struttura, tanto che le conformazioni di DNA a doppia elica fino ad ora evidenziate in laboratorio sono almeno una dozzina. Come illustrato[6] in Figura 1.3, tre di queste sono ritenute essere presenti in natura: A-DNA, B-DNA e Z-DNA. La forma B è quella originariamente descritta da James Dewey Watson e Francis Crick ed è quella predominante nell'ambito intracellulare.

La forma A è un'ampia spirale destrorsa, con il solco minore largo ma poco profondo e quello maggiore più stretto e profondo. Tale conformazione è

[5] Sono proteine che intervengono nella trascrizione da DNA a RNA. Si veda anche § 1.3.1

[6] ©Richard Wheeler. Immagine disponibile con licenza "*GNU Free Documentation license, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts*".

presente in condizioni non fisiologiche, quando il DNA viene disidratato, compiendo una rotazione ogni $11 bp$.

La forma B è la tipica conformazione riscontrabile dentro la cellula che corrisponde al modello di Watson e Crick. Questa spirale destrorsa compie un giro completo attorno all'asse ogni 10.4–10.5 paia di basi. Questa frequenza di rotazione, nota come *passo dell'elica*, dipende largamente dalle forze molecolari che ogni base esercita su quelle adiacenti.

La forma Z è tipica, invece, delle sequenze che presentano modificazioni chimiche come la *metilazione*[7], e dei tratti di DNA ricchi di basi C e G. Essa assume un andamento sinistrorso, opposto rispetto alla conformazione B. Ha il solco maggiore più superficiale e quello minore più stretto; deve il suo nome all'andamento a zig-zag che la caratterizza.

Inoltre, già in organismi semplici quali i batteri, vengono trovate strutture localmente a triplo filamento che si formano temporaneamente in concomitanza di particolari situazioni [1].

La quasi totalità degli organismi viventi immagazzina il patrimonio genetico in una o più molecole di DNA, dette *cromosomi*. I *procarioti*[8] posseggono un solo cromosoma, spesso di forma circolare; gli *eucarioti*[9] posseggono molteplici cromosomi, di lunghezza variabile, sempre lineari. Negli eucarioti i cromosomi assumono forma diversa a seconda della fase del ciclo cellulare: la *cromatina*, il materiale super-avvolto di acidi nucleici e proteine di cui sono costituiti i cromosomi, risulta essere una massa indistinta più espansa durante la fase stazionaria. Questa configurazione è necessaria perché l'informazione genetica possa esprimersi; mentre (si veda la Figura 1.4) durante la fase riproduttiva il nucleo scompare e

[7] La metilazione consiste nella sostituzione di un atomo di idrogeno con il *gruppo metile* a seguito di processi enzimatici. Questo fenomeno interviene nella regolazione dell'*espressione genica* (v. § 1.3.1) che converte l'informazione contenuta nei geni (segmenti di DNA) in macromolecole funzionali, quali proteine o RNA (v. § 1.3).

[8] I procarioti [dal greco *pro* 'prima' e *karyon* 'nucleo'], sono organismi unicellulari, tra cui i batteri, composti da un'unica cellula priva di nucleo ben differenziato o di altre suddivisioni interne; pertanto la molecola di DNA, circolare, si trova libera nel citoplasma.

[9] Gli eucarioti [dal greco *eu* 'bene' e *karyon* 'nucleo'], sono tutti gli organismi più evoluti, con nucleo ben definito e isolato dal resto della cellula tramite l'involucro nucleare, nel quale è racchiusa la maggior parte del materiale genetico.

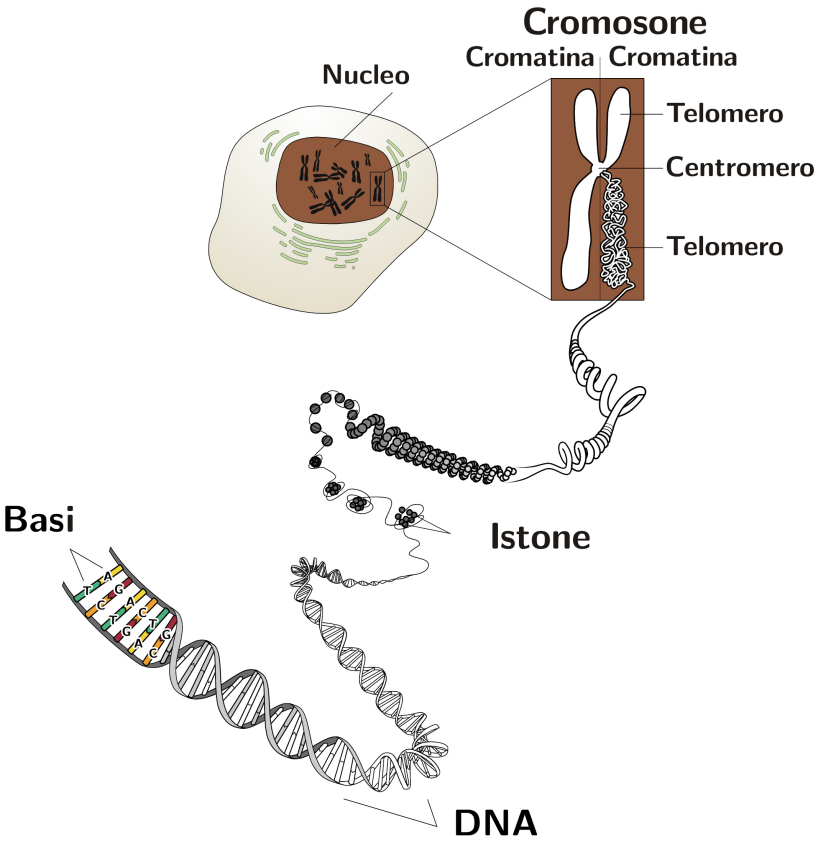


Figura 1.4: I successivi livelli di condensazione del DNA in cui si organizza la cromatina: partendo dalla doppia elica, la cromatina si costituisce come una matassa di DNA avvolto su istoni, per poi condensarsi sempre di più, fino ai cromosomi.

i cromosomi si impacchettano individualmente, assumendo la caratteristica forma a "X", con le varianti "V" e "Y".

La fase di nostro interesse è quella stazionaria, dato che, durante la riproduzione, la normale attività trascrizionale viene sospesa. La cromatina nucleare ha una struttura molto complessa e per molti aspetti ancora ignota: i filamenti che costituiscono i vari cromosomi sono mescolati insieme in modo apparentemente casuale, anche se misurazioni molto recenti sembrano indicare che cromosomi diversi tendano a occupare porzioni diverse del volume nucleare.

L'organizzazione finale della cromatina vede il DNA avvolto intorno a proteine (dette istoni) che lo costringono ad una super-spiralizzazione che permette

di compattarlo in modo altamente efficiente. Basti pensare che il genoma umano contiene un totale di 6×10^9 paia di basi: la sua dimensione lineare sarebbe compresa tra i 2,5m e i 3m, ed è completamente contenuto, insieme a molte altre molecole, in un nucleo cellulare del raggio medio di 3 μm.
Gli sforzi per comprendere meglio la struttura della cromatina sono supportati principalmente da due tecniche di indagine sperimentale, che permettono di rivelare la conformazione tridimensionale delle molecole biologiche: *risonanza magnetica nucleare* (NMR) e *diffrazione a raggi X* (X-Ray diffraction). Entrambe hanno un potere di discriminazione a livello atomico (0,4-0,2 nm), ma anche moltissime limitazioni che ne impediscono l'impiego su molecole grandi o di scarsa purezza. Di fatto, quindi, non sono applicabili a estensioni e complessità tipiche della cromatina nucleare. Si possono, però, osservare dei piccoli frammenti per dedurre conseguenze logiche da considerazioni generali di carattere chimico-fisico.

Per esempio, richiamando il concetto che C e G formano tre legami idrogeno, mentre A e T soltanto due, si comprende come zone del DNA molto ricche in G–C siano relativamente "rigide" e difficili da "fondere", ovvero risulta difficile separarne i filamenti. Al contrario, zone molto ricche in A–T si prestano con più facilità a separare i filamenti e dar vita a quelle strutture complesse a triplo o quadruplo filamento cui si è accennato sopra.

La capacità che il DNA possiede di assumere strutture tridimensionali peculiari è di fondamentale importanza per le molteplici funzioni biologiche che esso esplica: infatti, nonostante dal punto di vista strettamente chimico ogni molecola di DNA sia sostanzialmente uniforme, le funzioni biologiche che possono trovare sede in specifiche regioni sono molteplici e assai diverse tra loro. Presentiamo una limitatissima lista di regioni funzionalmente diverse.

1.3 Regioni codificanti

Le *regioni codificanti* sono i tratti del DNA che modellano le caratteristiche organiche di un essere vivente in quanto *espressione genica* del proprio patrimonio

ereditario. Tale processo, chiamato *sintesi proteica*, si compone di due grandi fasi:

trascrizione ovvero il processo mediante il quale le informazioni contenute nel DNA vengono trascritte in una molecola complementare di RNA[10].

traduzione durante la quale il filamento di RNA-messaggero, prodotto dalla trascrizione, è usato come stampo per la produzione di specifiche proteine: la formidabile relazione tra le basi dell'RNA e gli aminoacidi delle proteine è definita *codice genetico*.

Queste due fasi, a loro volta, contemplano un grande numero di passaggi, realizzati mediante il coinvolgimento di proteine, enzimi e varie strutture cellulari. L'informazione passa dal DNA alla proteina secondo un codice, scoperto da Nirenberg[11] nel 1961, che è sostanzialmente universale per gli organismi viventi e viene quindi detto *codice genetico standard*. Il patrimonio ereditario (v. Figura 1.5) è scritto nella catena del DNA con il codice genetico, il quale, trascritto in RNA, mette in corrispondenza le basi azotate con gli aminoacidi. Ciascuna parola del codice è costituita da una serie di tre basi di RNA, e per questo è noto anche come codice "a triplette" .

Durante la sintesi proteica ognuna delle triplette[12] sull'RNA indica esattamente quale aminoacido verrà inserito nella catena proteica che si sta costruendo. Da ciò si coglie che il fenomeno genetico fondamentale, a livello cellulare, è la sintesi delle proteine.

Un *gene* consiste di una lunga sequenza di DNA che contiene regioni codificanti e non-codificanti più vari segmenti di controllo. Le regioni codificanti determinano

[10] La sintesi dell'*acido ribonucleico*, o RNA, procede utilizzando la molecola di DNA come impronta o *template* (v. § 1.3.1). La catena polinucleotidica di RNA differisce da quella di DNA perché: (1) contiene lo zucchero *ribosio* anziché il deossiribosio; (2) la timina (T) è sostituita dall'*uracile* (U); (3) si ritrova prevalentemente nella forma a singolo filamento.

[11] Nobel per la medicina 1968.

[12] La tripletta di basi di RNA prende il nome di *codone*: essendo disponibili quattro basi (ACGU), esistono dunque $4^3 = 64$ codoni diversi; gli aminoacidi che entrano nel processo sono molti meno perchè diversi codoni sono *ridondanti* ovvero, due o più codoni corrispondono allo stesso aminoacido

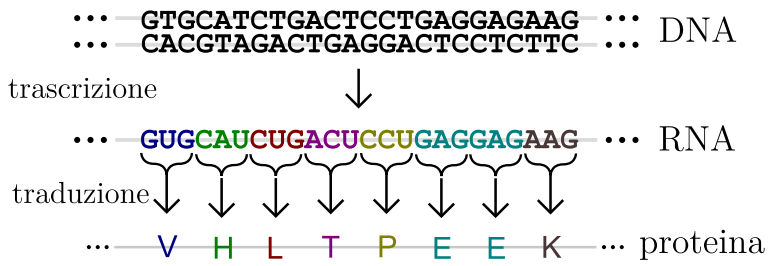


Figura 1.5: Processo schematico in cui si evidenzia la trascrizione dal DNA all'RNA e la traduzione, indotta dal codice genetico, da RNA in sequenza di aminoacidi della proteina.

cosa il gene produce, mentre quelle non codificanti possono regolare le condizioni dell'espressione del gene. Qualsiasi cellula dell'organismo umano possiede lo stesso genoma[13]: è l'espressione differenziale dei geni che rende, ad esempio, una cellula muscolare diversa da un neurone, oppure permette lo sviluppo di un feto a partire dalle cellule embrionali.

Negli eucarioti, le regioni dei geni che partecipano alla traduzione in proteine sono intervallate da zone che non verranno tradotte: i segmenti rimossi dalla traduzione prendono il nome di *introni*, mentre le zone tradotte si chiamano *esoni*. Gli introni si differenziano da altri segmenti di DNA realmente non codificante, dal momento che possono 'attivarsi' e diventare esoni, in dipendenza di molteplici fattori, come per esempio il tipo di tessuto in cui si esprime il gene. Il gene eucariota ha dunque una struttura modulare, e il contesto ne determina il ruolo effettivo: moduli che sono esoni in un contesto possono invece essere introni in un altro.

Occorre sottolineare come l'informazione che codifica la proteina sia custodita su uno solo dei due filamenti – senso o antisenso – ed esistano geni su entrambi. Dal punto di vista dell'espressione genica, l'informazione contenuta nei filamenti complementari è indipendente. In altre parole, traducendo l'informazione dal

[13]Il genoma di un organismo è l'intera sequenza genetica presente nei cromosomi, includendo sia i geni sia le sequenze non codificanti del DNA.

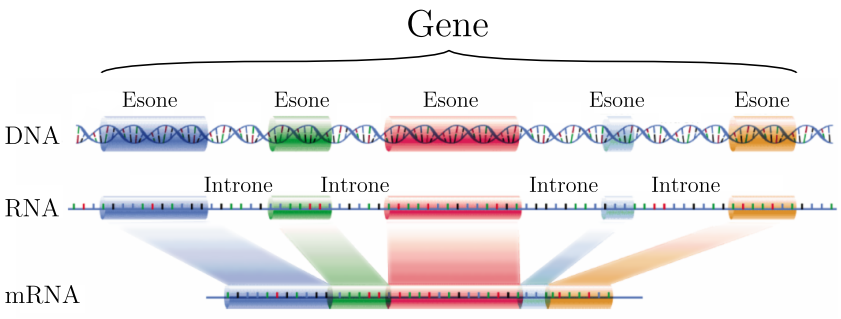


Figura 1.6: Nella fase di trascrizione da DNA a RNA-messagero avviene una scrematura in cui vengono rimossi gli introni prima della successiva traduzione degli aminoacidi.

filamento complementare si otterrebbe una proteina del tutto diversa da quella normalmente specificata. Questo è uno degli aspetti spesso sottovalutati quando si rappresenta il DNA come se fosse a singolo filamento.

Da un punto di vista informatico, le regioni codificanti sono caratterizzate da una distribuzione quasi uniforme delle 4 basi, con una leggera correlazione sulla terza base e, per massimizzare la stabilità strutturale, in caso di possibilità di scelta, è preferita una base in grado di instaurare legami più forti quali C o G.

1.3.1 Regioni regolatrici

I geni presenti nel genoma non sono tutti attivi simultaneamente: alcuni vengono attivati solo in fasi specifiche dell'esistenza dell'organismo (per es., sviluppo embrionale), altri solo in risposta a determinati stimoli esterni. Questi mutamenti funzionali, nel complesso, prendono il nome di *regolazione dell'espressione genica*. I meccanismi che regolano l'espressione genica sono molteplici e qualitativamente diversi: il meccanismo può essere un impacchettamento differenziale che rende regioni molto compresse inaccessibili e lascia i geni ivi contenuti non espressi, oppure anche un intervento attivo di specifiche proteine.

Quest'ultimo caso è sicuramente il più interessante perché è quello su cui si basa la normale regolazione dell'attività cellulare. Uno schema generale, illustrato

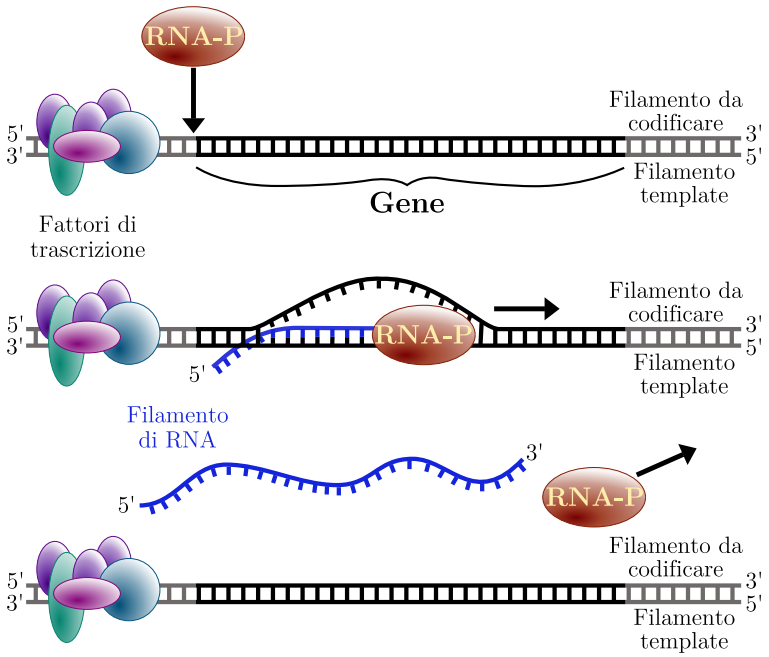


Figura 1.7: Dall'alto verso il basso è visibile il processo schematico di trascrizione del DNA. L'enzima RNA-Polimerasi può iniziare la copia del gene solo dopo aver individuato l'agglomerato di fattori di trascrizione.

in Figura 1.7, di questo tipo di meccanismo regolatore prevede l'esistenza di una regione, detta *promotore*, localizzata a monte del gene da trascrivere, sulla quale si legano delle proteine dette *fattori di trascrizione* (TF, *transcription factors*). Le esatte sequenze promotrici cui i fattori di trascrizione TF si attaccano vengono denominate *siti di congiunzione per fattori di trascrizione* (TFBS, *transcription factor binding site*). Quando i TF si legano ai TFBS, la struttura locale del DNA viene modificata e il gene soggetto a regolazione diventa disponibile a venir trascritto enzimaticamente[14].

Un aspetto molto importante di questo meccanismo è che il riconoscimento dei siti di congiunzione TFBS da parte dei TF non è solamente basato sulla particolare sequenza di basi nella regione, ma anche sulla struttura tridimensionale che tale sequenza assume: l'adesione del fattore di trascrizione al sito di congiun-

[14]L'enzima *RNA-polimerasi* provvede a copiare la sequenza di DNA per produrre una striscia di RNA-messaggero, così chiamato perché trasporta il messaggio genetico dal DNA verso il macchinario della sintesi proteica cellulare.

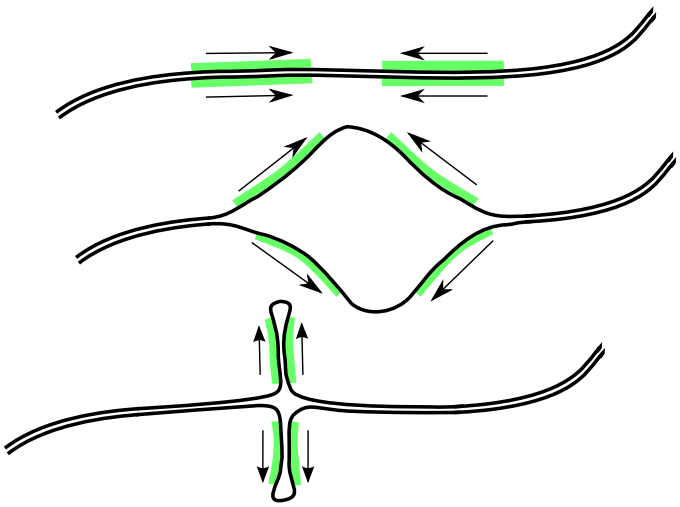


Figura 1.8: Complessa struttura a "doppia forcina". I segmenti evidenziati sono "*inverted repeats*" che rendono possibile questo tipo di configurazioni.

zione sul DNA è un incastro geometrico. Come già visto nel § 1.2, la struttura 3D assunta dal DNA può essere molto complessa e dipende dalle basi presenti in quella sezione: un esempio è mostrato nella Figura 1.8.

Quando due sequenze poste a distanza ravvicinata sono tali che l'una è la complementare invertita dell'altra – *inverted repeats* – la struttura tridimensionale del DNA oscilla tra due forme in equilibrio: una di tipo "lineare", la classica doppia elica, e l'altra "cruciforme", o "a doppia forcina", che è strutturalmente molto complessa. Organizzazioni simili a questa sono molto frequenti nel DNA e rappresentano il vero bersaglio di riconoscimento per i fattori proteici che al DNA si devono legare.

Ricerche a priori, usando approcci algoritmici orientati a identificare tutte le zone in cui sono possibili simili organizzazioni strutturali, a prescindere dall'esatta sequenza di lettere occorrente, hanno portato all'individuazione, nel genoma dello stesso organismo, di altre possibili sequenze candidate a siti di congiunzione per fattori di trascrizione TFBS.

1.4 Regioni ripetute

Le ripetizioni, dirette o invertite, assumono un'importanza fondamentale nella genomica, perché ripetizioni di segmenti occorrono in una grande varietà di circostanze. Una minima classificazione comprende alcune grandi sottoclassi illustrate di seguito.

1.4.1 Ripetizioni non tradotte del mRNA e patologie

Le ripetizioni nelle regioni non tradotte (UTR, *Untraslated regions*) dell'RNA-messaggero si presentano come ripetizioni in tandem in cui un singolo nucleotide, tipicamente poli A, si presenta decine di volte in piccoli gruppi di lettere (coppie o triplette). Tali ripetizioni, seppur escluse dall'espressione genica, esercitano però una qualche regolazione del processo di traduzione del mRNA in proteina, influenzando quindi il tasso di produzione della proteina e, in relazione al numero di ripetizioni, anche la sua composizione.

Simili strutture ripetute possono figurare anche all'interno di introni, e sono spesso associate a sindromi di genetica dinamica, ovvero a patologie la cui insorgenza o intensità varia con il progredire delle generazioni. Il decorso di tali patologie ereditarie consiste in un aumento (o diminuzione) costante del numero di nucleotidi sotto esame: supponiamo, per esempio, che il soggetto normale mostri 11 ripetizioni di uno stesso appaiamento di basi; se ad ogni generazione aumenta di 3 il numero di ripetizioni, allora avremo la prima generazione con 14 paia, la seconda con 17 e la terza con 20.

L'intensità e la rapidità di insorgenza della patologia è proporzionale al numero di ripetizioni presenti, anche se il meccanismo non è ancora stato compreso del tutto. In questo caso, spesso si tratta di ripetizioni esatte.

1.4.2 Ripetizioni di geni

Accade frequentemente che geni molto simili si presentino in cluster ravvicinati: questo perché spesso geni simili derivano da duplicazioni di geni ancestrali e la localizzazione del nuovo gene è in tandem col vecchio. In questo caso, spesso la

duplicazione comprende tutta la struttura del gene, compresi esoni, introni ed eventuali sequenze regolatrici. Si tratta di ripetizioni con errore (o ripetizioni approssimate) dal momento che il processo di duplicazione non è esatto, e che la selezione naturale e l'evoluzione nel suo complesso favoriscono l'accumularsi di mutazioni che differenziano l'originale dalla copia.

1.4.3 Elementi mobili ed esogeni del DNA

Il DNA eucariota ospita al suo interno molteplici segmenti *esogeni*, ovvero appartenenti ad altri corredi genetici, e integrati in tempi evolutivamente molto lontani. Tra essi, un ruolo di primaria importanza è ricoperto dai *trasposoni*[15], elementi mobili del DNA, spesso di origine virale. Con il termine mobile si intende dire che questi elementi sono in grado di cambiare localizzazione all'interno del DNA, saltando su regioni diverse dello stesso cromosoma, o addirittura su altri cromosomi. I trasposoni a volte, invece di spostarsi, si duplicano in altre regioni del genoma.

Figura 1.9: Le variegate cariossidi del mais: circa il 50% del genoma del mais è composto da trasposoni.

Chiaramente (vedi Figura 1.9), queste operazioni di inserzione possono provocare effetti più o meno gravi a seconda della particolare destinazione in cui il

[15]I primi trasposoni furono scoperti da Barbara McClintock nel 1948, mentre studiava le variegate colorazioni delle cariossidi (chicchi) del mais. Per la scoperta ricevette il premio Nobel nel 1983.

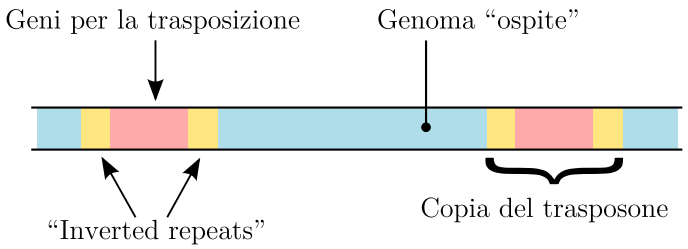


Figura 1.10: Due trasposoni batterici identici le cui estremità risultano invertite e complementari (“inverted repeats”).

trasposone si integra: se cade all’interno di un gene, per esempio, può alterarne la proteina prodotta o modificarne radicalmente l’attività, disattivando il gene o, in casi estremi, attivando geni inattivi.

I trasposoni sono di diversa tipologia e mostrano lunghezze variabili da poche centinaia ad alcune migliaia di basi. Anche il numero delle loro ripetizioni varia moltissimo, ma è comunque assai elevato e, per alcune famiglie, può raggiungere le centinaia di migliaia di copie. Ad una stima attuale, oltre il 30% del genoma umano (e il 50% di alcune specie vegetali) sembra composto di elementi trasponibili di diversa lunghezza e numero di copie [2].

Un particolare molto interessante di alcuni trasposoni è che il meccanismo di “*cut-and-paste*” con cui riescono a rilocalizzarsi sul DNA comprende la presenza, agli estremi del trasposone, di coppie di sequenze complementate e invertite (v. § 1.3.1). La ricerca di trasposoni, quindi, può tener conto sia di lunghe ripetizioni dirette, sia della compresenza di piccole ripetizioni complementate e invertite.

1.4.4 Regioni di attacco alla matrice cromatinica

Le *Matrix attachment regions* o anche *Scaffold/matrix attachment regions* (MAR o S/MAR, letteralmente impalcatura/regione d’ancoraggio alla matrice [cromatinica]), sono sequenze tipiche del DNA eucariota e corrispondono a segmenti che

si legano alle fibre interne al nucleo cellulare fornendo quindi, alla cromatina, dei punti di ancoraggio per la formazione di particolari strutture.

La loro localizzazione concorre alla regolazione dell'espressione genica. Le MAR sono spesso identificate *in vitro*, quando frammenti di DNA vengono posti a contatto con proteine della matrice cromatinica e si osserva l'istaurarsi di un legame in una particolare zona del frammento. Per similitudine delle condizioni chimico-fisiche, si presuppone che la stessa cosa avvenga anche *in vivo*, all'interno del nucleo cellulare. Le S/MAR sono regioni di lunghezza variabile, da 100 a 1000 bp, non riferibili a vere sequenze consenso, quindi non identificabili coi metodi classici di *pattern matching*; nell'uomo ne sono state stimate circa 300.000.

Molte MAR sono anche *siti di origine di replicazione* (ORI, abbr. di *origin of replication*) ovvero le zone dove ha inizio la replicazione del DNA: è significativo che, nell'uomo, il numero delle MAR che sono anche ORI è stimato intorno ai 30 mila, che è anche il numero stimato di geni. Le MAR non contengono vere e proprie ripetizioni, ma piuttosto pattern di motivi variamente conservati, che vedono spesso una presenza molto alta delle basi A e T. La cosa è in linea con la necessità che, in queste regioni, il DNA possa essere facilmente aperto e possa assumere conformazioni peculiari.

La struttura tridimensionale non è ancora stata descritta, ma diverse evidenze indicano che il DNA di una MAR può organizzarsi anche in *mispaired segments*, ovvero frammenti in cui gli appaiamenti (almeno apparentemente) non sono perfetti, formando anche coppie usualmente non ammesse (quali A–C) [3]. Da un punto di vista di ricerca testuale, è molto probabile che sequenze MAR possano ospitare permutazioni esatte (o permutazioni con pochi errori) di una determinata sequenza elementare.

1.4.5 Siti fragili e regioni altamente ripetute

Siti fragili e regioni altamente ripetute mostrano alcune analogie con le regioni MAR. Al pari di esse sono spesso ricche di basi A e T, e mostrano un'organizzazione strutturale certamente complessa e ancora ignota. I siti fragili sono

relativamente corti (meno di 100 bp) e sono punti di frequente rottura del DNA (soprattutto in condizioni patologiche); le rotture che si verificano durante la replicazione portano a gravi conseguenze, come cancellazioni di geni singoli o a gruppi, mutazioni cromosomiche, ecc.

In questo contesto, le regioni altamente ripetute sono costituite da pattern di lunghezza variabile (da 10 a 1000 bp) che si ripetono un numero molto elevato di volte (nell'ordine delle centinaia di migliaia). Anch'esse sono prevalentemente ricche in A e T, e sono strutturalmente associate a due zone precise del cromosoma in fase riproduttiva: il *centromero*, ovvero, nei cromosomi a "X" è il centro, e i *telomeri*, ovvero le estremità della X (v. Figura 1.4 a pag. 7). In queste zone, la ricchezza in A e T, unita alla frequente ripetizione, anche se non completamente esatta, di motivi che possono anche essere palindromi (occorrenza diretta più la complementata invertita) generano plausibilmente strutture tridimensionali molto complesse: successioni di forme cruciformi mutuamente sovrapposte ecc.

Anche in questo caso, una ricerca per segmenti permutati potrebbe dare grossi risultati e riuscire a descrivere il modulo unitario che si ripete, in quanto l'eventuale rumore puntiforme potrebbe agevolmente rientrare nella descrizione a permutazioni.

1.5 Indagini algoritmiche

Il sequenziamento dei genomi di molti organismi (639 completati e 3000 in corso di completamento al Settembre 2007 [4]) ha portato ad accumulare una quantità di dati impensabile solo fino a poco tempo addietro. Il problema maggiore consiste nell'estrarre, da questa massa di dati, informazioni rilevanti dal punto di vista biologico. Tale operazione non risulta affatto facile: le tecniche moderne di sequenziamento, infatti, non producono alcuna informazione circa le proprietà biologiche della regione sequenziata. D'altro canto, se sequenziare un genoma è diventata un'operazione relativamente facile e veloce, acquisire informazioni sulle proprietà biologiche associabili alle varie regioni del genoma è una questione ancora aperta: i meccanismi molecolari alla base di diverse funzioni cellulari sono

infatti spesso ignoti, o noti solo in parte. Di conseguenza, la descrizione funzionale del genoma si arricchisce di particolari, anche qualitativamente nuovi, man mano che le ricerche procedono: per questo i vari database biologici pubblicano aggiornamenti periodici (in genere trimestrali) in cui la parte più variabile è rappresentata dalle annotazioni, ossia dai meta-dati che specificano informazioni circa ruoli e proprietà svolti da determinate sequenze.

Gli approcci sperimentali per l'attribuzione di funzioni biologiche a sequenze genomiche sono molti e potenti, ma davanti all'estensione di un genoma medio, mostrano un'inevitabile inadeguatezza: richiedono tempi molto lunghi. È questo il problema principale dell'era post-genomica: la forbice che si va creando tra l'accumulo dei dati "grezzi" e l'acquisizione di informazioni circa il ruolo biologico da loro ricoperto, che è poi l'aspetto nodale della ricerca.

In questo contesto, l'adozione di approcci teorici a priori (come per esempio la ricerca di tutte le ripetizioni approssimate di sequenze con lunghezza minima fissata e margine d'errore massimo fissato), o basati su *machine learning approaches* (algoritmi adattativi che eseguono pattern recognition o clustering sulla base di un training set di casi noti), potrebbero permettere uno screening veloce di grandi masse di dati restringendo la scelta di sequenze candidate per l'attribuzione a un ruolo specifico.

Il problema risulta molto complesso, in primo luogo perché gli iniziali criteri desumibili dalle ricerche genomiche sono scarsamente attendibili e, dunque, non adatti per impostare le ricerche a priori o machine learning. La genomica, infatti, è una scienza molto giovane e la probabilità che scoperte prossime stravolgano impianti o criteri esistenti è molto alta.

L'altro problema fondamentale risiede proprio nella natura dei dati: i genomi, soprattutto quelli eucarioti, sono quanto di più dissimile da una collezione di dati omogenei e ordinati. Come abbiamo mostrato in questa rapidissima rassegna, a dispetto della uniformità della chimica di base, il DNA è un groviglio di sequenze diversissime per struttura e funzione biologica.

Un'immagine molto efficace della situazione in cui si viene a trovare chi voglia cimentarsi nell'impresa di attribuire un ruolo alle sequenze genomiche, risale al

1992, e mette in analogia le sfide poste dal Progetto Genoma Umano con il problema di comprendere cosa contenga un enorme disco fisso di un sistema sconosciuto. Ottenere la sequenza corrisponde ad ottenere un'immagine del disco, mentre comprenderne il significato è equivalente a dover fare *reverse engineering* a partire da un sistema sconosciuto per arrivare, tutto all'inverso, fino a dedurne le specifiche architetturali e operative [5].

A distanza di anni, i termini del problema e il contesto non sono variati in modo sostanziale: semplicemente oggi disponiamo di un numero maggiore di dati, di criteri e di algoritmi per affrontare le ricerche.

I maggiori risultati ottenuti con l'approccio a priori sono stati conseguiti cercando all'interno dei genomi tratti ripetuti di lunghezza diversa e con diverso margine di errore, con il risultato di scoprire alcune sequenze regolatrici, i cluster di geni, il DNA altamente ripetuto, e altre sequenze di significato biologico. Un altro approccio fruttuoso è stato quello di cercare motivi strutturati, che, come abbiamo visto, può portare all'identificazione di TFBS e di altri elementi funzionali, come per es. i trasposoni, caratterizzati dal motivo del doppio palindromo alle estremità.

La ricerca per permutazioni potrebbe permettere di individuare, con approcci a priori, altre sequenze di significato funzionale: risulta indispensabile definire un criterio per la ricerca di sequenze identiche a meno di permutazioni, eseguirne la ricerca all'interno di una regione di genoma ricca di *features* e annotazioni, in modo da avere speranza di porre in corrispondenza i risultati della ricerca con qualche specifica feature.

Capitolo 2

Analisi di sequenze genomiche

Dal 1992, l'anno in cui è stata lanciata la sfida, il Progetto Genoma Umano ha portato con sé grandi speranze e qualche nuova paura, ma, soprattutto e ancor prima di queste, una enorme quantità di dati che devono ancora essere sottoposti ad analisi per inferirne nuove informazioni e conoscenze.

Lo studio del genoma comporta il sequenziamento del DNA, cioè l'identificazione della sequenza dei 6 miliardi di coppie di basi azotate che ne compongono la molecola (v. § 1.2). La comprensione della funzione dei geni e di quali malattie possono derivare dalle loro alterazioni costituisce l'obiettivo finale del progetto.

La tecnica classica per l'annotazione genomica, fa uso dei concetti biologici di geni *ortologhi*[1] o *paraloghi*[2]: per decidere se due geni sono simili si ottiene informazione stabilendo relazioni ortologhe o paraloghe con altri geni ben caratterizzati in altri organismi.

Lo sforzo scientifico si è gradualmente organizzato mettendo in campo sempre più spesso équipe composte da biologi, matematici ed informatici, con l'obiettivo comune di affrontare il problema da più parti e con prospettive diverse. Inoltre

[1] Due sequenze sono ortologhe se sono state separate da un evento di *speciazione*: quando una specie diverge in due specie diverse, le copie divergenti di un singolo gene nelle sottospecie sono detti ortologhe. I geni ortologhi, sono geni in specie differenti che risultano simili fra loro perché sono originati da un antenato comune.

[2] Due sequenze sono paraloghe (v. § 1.4.2) se sono separate da un evento di duplicazione genica: quando un gene in un organismo viene duplicato andando ad occupare due diverse posizioni nello stesso genoma, allora le due copie sono paraloghe.

l'ampia disponibilità di genomi completamente sequenziati per un gran numero di organismi viventi [4] schiude nuove possibilità d'applicazione delle tecniche di *information retrieval* per il confronto di interi genomi: il *pattern* o *motif discovery* nei dati è largamente usato come un mezzo per 'comprendere' grossi volumi di dati quali quelli delle sequenze di DNA. L'analisi delle sequenze è impostata sulla base di due aspetti differenti:

- Nel genoma sequenziato è stata rilevata, oltre ai circa 30.000 geni, un'enorme quantità di sequenze di cui ancora non si conoscono funzionamento e scopo. A seconda delle possibili considerazioni, la maggior parte dell'intera sequenza viene reputata non codificante (v. § 1.3.1), o "*DNA-spazzatura*", in una stima che varia dal 72% al 98% [6]. Sono dunque molti gli sforzi, come anche questa tesi, che si concentrano sull'analisi della sequenza di nucleotidi.

- Nel campo della comparazione genomica ad alto livello [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17] – ovvero considerando i geni piuttosto che i nucleotidi come l'entità più piccola – l'attenzione è rivolta all'ordine e al contenuto dei geni: la speciazione si può "misurare" come differenza fra i vari genomi. Dunque, le regioni di geni che preservano un certo ordine nel loro raggruppamento, forniscono indicazioni su insiemi funzionalmente associati di geni [9, 10].

Dal momento che non esiste ancora sufficiente conoscenza per costruire un adeguato modello per filtrare permutazioni di caratteri rilevanti da altre apparentemente senza significato, spesso si ricorre ad approcci che non prevedono l'uso di un modello di dati [18].

In questo capitolo[3] presentiamo il connubio allo stato dell'arte fra la genomica e l'informatica. Esistono molte proposte teoriche, in ambito accademico, per quanto riguarda lo studio dei modelli formali per le analisi genomiche (capitoli 2.1 – 2.5) ma, allo stesso tempo, quotidianamente vengono fornite nuove rispo-

[3]Continueremo ad usare in modo intercambiabile le parole geni e genomi come esempi specifici di una trattazione relativa a concetti più generali su stringhe di simboli.

ste usando applicativi esistenti che pongono, ragionevolmente, l'enfasi più sulla rapidità della risposta che sulla eccessiva precisione (§ 2.6).

Prima introduciamo però alcuni concetti formali [19] preliminari alla migliore comprensione delle soluzioni esposte.

2.1 Rappresentare il Genoma

La sequenziazione del DNA produce una lunga successione di simboli che rappresentano gli elementi chimici sulla molecola. Intuitivamente possiamo assumere che tale successione sia una sequenza delle lettere {A, C, G, T}. Segmenti di questa sequenza vengono raggruppati in unità più grandi che prendono il nome di geni. Ad ogni gene viene assegnato un nome[4] e il genoma può essere visto come una successione dei nomi di geni.

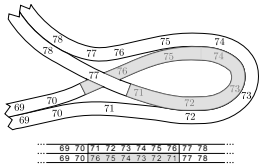
Confrontare il contenuto di geni o determinarne l'ordinamento relativo sono operazioni imprescindibilmente basate sulla nostra capacità di decidere quando due geni sono "uguali". Questo non è affatto un compito banale e, dalla scoperta delle prime ricombinazioni cromosomiche[5], sono stati tentati numerosi approcci. Le tecniche più attendibili di identificazione genica si basano sugli allineamenti di sequenze, sia di nucleotidi in sé, oppure dei corrispondenti aminoacidi (v. § 1.3).

Stabiliti i criteri biologici per definire quali geni sono uguali, confrontare l'ordinamento dei geni può essere ridotto al confronto fra stringhe di simboli orientati (*signed letters*): infatti, dal momento che i due filamenti appaiati sono antiparalleli e complementari (v. § 1.1), è possibile rappresentare l'informazione genetica come una stringa di lettere

$$(\; a \quad b \quad -c \quad d \quad -e \quad f \;)$$

[4] I nomi assegnati ai geni, meno intuitivamente, combinano serie di numeri e lettere del dominio biologico.

[5] Nel 1938 Dobzhansky e Sturtevant osservarono [20] inversioni di ampi segmenti cromosomici come quelle mostrate in questa figura.



in cui le lettere identificano i geni e i segni negativi indicano un'orientazione inversa. In particolare, per due stringhe a e b che rappresentano DNA, si definisce la relazione di uguaglianza:

$$\begin{aligned} a &= a_1 a_2 \dots a_n \\ b &= b_1 b_2 \dots b_n \end{aligned}$$

$$a = b \Leftrightarrow \begin{cases} a_i = b_i \\ \text{oppure} \qquad\qquad i = 1 \dots n \\ a_i = -b_{n-i+1} \end{cases} \tag{2.1}$$

Fissato un insieme di geni Σ, quando due genomi sono rappresentati da stringhe che contengono esattamente un'occorrenza di ciascun gene in Σ, qualuque sia la sua orientazione, allora chiamiamo tali stringhe *permutazioni*.

Esempio 2.1. *Sia dato l'insieme di geni* $N = \{a, b, c, d, e, f\}$*; le stringhe*

$$G_1 = (\ a \quad b \quad -c \quad d \quad -e \quad f\ ) \text{ e } G_2 = (\ -a \quad d \quad -e \quad b \quad -c \quad f\ )$$

sono permutazioni di Σ*. Mentre*

$$G_3 = (\ a \quad b \quad -c \quad a \quad d \quad -e \quad f\ ) \text{ e } G_4 = (\ -a \quad d \quad -e \quad -c \quad f\ )$$

non sono permutazioni poiché il gene a *è duplicato nel genoma* G_3 *e il gene* b *manca dal genoma* G_4*.*

Scegliere di rappresentare i genomi come permutazioni oppure sotto forma di stringhe con elementi duplicati presenta delle implicazioni sugli aspetti biologici e computazionali dell'analisi genomica. Di fatto, la complessità computazionale degli algoritmi per stringhe con ripetizioni è più elevata rispetto a quelli per le permutazioni. Allo stesso tempo, da un punto di vista biologico, è un'ipotesi molto forte assumere che non vi siano ripetizioni di geni in un insieme di genomi. Pertanto, rappresentare i genomi come insiemi di permutazioni richiede un'analisi preliminare che risolva le ambiguità dovute ai geni duplicati, in taluni casi ricorrendo alla rimozione dei duplicati [21, 22, 23].

2.2 Cluster di geni

Analizzando l'ordinamento fra i geni in più genomi si scopre che molto difficilmente questi ordinamenti rimangono identici (v. § 1.4), anche per specie molto affini [24] o addirittura su diversi filamenti dello stesso batterio [25]. Tuttavia queste disposizioni non sono casuali e si riscontrano segmenti genomici che presentano lo stesso contenuto genetico, con ordinamenti *simili* dei geni. Tali segmenti sono chiamati *conserved gene clusters* o semplicemente cluster di geni. Ai fini della trattazione è sufficiente una definizione informale:

Definizione 2.1 (**Cluster di geni** – informale)**.** *Un cluster di geni è un* insieme di geni *che, per ragioni biologiche, si sono mantenuti "più o meno ravvicinati"* in uno stesso segmento *anche su genomi diversi.*

Da un punto di vista combinatorio si vuol porre l'attenzione sulle differenti nozioni usate per definire un cluster, che è un *insieme* – senza ordinamento interno – distribuito su un *segmento*, ovvero una stringa totalmente ordinata.

Le relazioni utili che possono intercorrere fra insiemi di geni e segmenti del genoma sono chiarite con alcuni esempi. A questo punto, non si fanno ancora assunzioni particolari riguardo alla rappresentazione che può essere sia la permutazione o la stringa con ripetizioni.

Esempio 2.2 (**Numero di insiemi**)**.** *Come si comportano insiemi di lettere su stringhe o permutazioni? Evidenziamo le occorrenze degli elementi dell'insieme* $\{e, g, n\}$ *nella frase*[6]*:*

"il gene del genoma sottointende un cromosoma"

Sembra risaltare, per le parole "geni" e "genoma", una comune funzione semantica. Considerando un nuovo insieme $\{c, s, t\}$*, però, non si ricavano informazioni utili, almeno apparentemente:*

"il gene del genoma sottointende un cromosoma"

[6] Si chiede venia al lettore per la ridicola poetica.

È possibile continuare con altri esperimenti, più o meno utili, ma è già evidente che abbiamo un *numero esponenziale di insiemi* pari alla cardinalità dell'insieme delle parti dell'alfabeto A: $\mathcal{P}(A)$.

Esempio 2.3 (***Occorrenza* di un cluster**). *Evidenziando l'insieme* $\{e, g, n\}$ *si ottiene dunque:*

"il gene del genoma sottointende un cromosoma"

Si noti la vicinanza fra gli insiemi individuati in "sottointende" – sono separati da singole lettere "d" e "t". Questo potrebbe suggerirci di unire quelle occorrenze così vicine, diminuendo il numero di cluster individuati, a patto di ammettere degli intrusi *nei nuovi cluster trovati: "nten" , "ende" o addirittura "ntende".*

La decisione di accettare degli elementi estranei, mantenendo ferma l'attenzione sui cluster più che sugli intrusi, ci permette di modellare dei *gap* all'interno dei nostri cluster. Questo potrebbe essere biologicamente ammissibile considerando che i geni possono essere persi o acquisiti (v. § 1.4.2 e § 1.4.3).

Esempio 2.4 (**Estensione di un insieme**). *Evidenziando ora l'insieme* $\{a, o, n, s\}$*, si ottiene:*

"il gene del genoma sottointende un cromosoma"

Nelle parole "genoma" e "cromosoma" le due occorrenze dell'insieme dato con un gap di dimensione $\delta = 1$ *sono le sottostringhe "noma" e "omosoma".*

La presenza della lettera "m" in entrambe le occorrenze potrebbe suggerire di *estendere* l'insieme ad uno più significativo {a, o, m, n, s}.

I tre esempi descrivono adeguatamente le osservazioni sperimentali di geni funzionalmente correlati, anche in genomi diversi, che appaiono spesso nelle reciproche vicinanze. Ciò può dipendere da molti fattori come ereditarità evolutiva o selezione funzionale [17].

È possibile specificare diversi modelli formali imponendo opportuni vincoli sui sottoinsiemi ammissibili di geni, sul tipo di stringhe considerate e sul numero e la natura delle occorrenze.

2.3 Intervalli comuni

Un primo modello dei cluster di geni si ottiene assumendo che i genomi siano permutazioni reciproche e richiede che le occorrenze dei cluster siano uguali, usando la nozione di uguaglianza (2.1) indicata a pagina 24. In questo caso si parla di segmenti comuni o *conserved segments*.

2.3.1 Intervalli comuni su permutazioni

Gli intervalli comuni sono la prima generalizzazione dei *conserved segments*, ottenuti rilassando i vincoli che i geni appaiano nello stesso ordine e con lo stesso orientamento. Apparsi in letteratura [7] applicati a due sole stringhe, sono stati successivamente estesi a K stringhe. Dal momento che l'orientazione non è necessariamente preservata, si ignorano i segni delle lettere.

Definizione 2.2 (Intervalli comuni su permutazioni)**.** *Sia* Π *un insieme di permutazioni sull'alfabeto* Σ*. Un sottoinsieme* $I \subseteq \Sigma$ *è un* intervallo comune *se presenta un'occorrenza, senza gaps, in ciascuna permutazione di* Π*.*

Intuitivamente un intervallo comune è un insieme di elementi che appaiano in qualunque ordine, ma consecutivamente, in ciascuna delle permutazioni.

Siano le due permutazioni:

$$G_1 = (\ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11\ )$$

e

$$G_2 = (\ 4 \quad 2 \quad 1 \quad 3 \quad 7 \quad 8 \quad 6 \quad 5 \quad 11 \quad 9 \quad 10\ )$$

Si noti che se una delle due permutazioni è l'identità, come per G_1, allora gli intervalli comuni sono insiemi di interi consecutivi. Se due intervalli comuni hanno un'intersezione non banale, come per esempio {1, 2, 3, 4, **5**, **6**, **7**, **8**} e {**5**,

6, **7**, **8**, 9, 10, 11}, allora l'intersezione è ancora un intervallo comune poiché, per definizione, è un intervallo di entrambe le permutazioni.

Proposizione 2.1. *Siano S e T due intervalli comuni sull'insieme di permutazioni Π. Se S e T risultano sovrapposti, allora tutti gli intervalli $S \cup T$, $S \cap T$, $S \setminus T$ e $T \setminus S$ sono intervalli comuni.*

Come è possibile intuire dalla proposizione, il numero di intervalli comuni di un insieme di permutazioni di n elementi è $\mathcal{O}(n^2)$, limite facilmente ottenibile comparando due permutazioni identiche. Questo aspetto rende poco maneggevoli i risultati ottenibili su sequenze biologiche reali.

Risultati migliori si ottengono usando intervalli comuni irriducibili:

Definizione 2.3 (Intervalli comuni irriducibili)**.** *Sia Π un insieme di permutazioni, un* intervallo comune irriducibile *è un intervallo comune che* non *è unione di altri intervalli comuni. L'unione di intervalli comuni è un intervallo comune* riducibile.

Sia Π un insieme di permutazioni di lunghezza n, le caratteristiche associate agli intervalli comuni irriducibili sono:

- il numero di intervalli comuni irriducibili di Π è $\mathcal{O}(n)$;
- unendo gli intervalli irriducibili si generano tutti gli intervalli comuni di Π.

2.3.2 Intervalli comuni su stringhe

Analizziamo quali rilassamenti e quali vincoli sono imposti alla definizione generale di cluster di geni per ottenere gli intervalli comuni su stringhe:

Definizione 2.4 (Intervalli comuni su stringhe)**.** *Sia $\mathcal{G}$ un genoma composto da stringhe sull'alfabeto Σ. Un sottoinsieme $I \subseteq \Sigma$ è un* intervallo comune *se presenta un'occorrenza, senza gaps, in ciascuna stringa di $\mathcal{G}$.*

Questa definizione è identica a quella per le permutazioni 2.2 eccetto per una parola: "stringa" invece di "permutazione". Questo modello è particolarmente

adatto allo studio di stringhe con molte ripetizioni, in cui i cluster di geni sono stati ricombinati da eventi di duplicazione, interni, producendo svariate copie dello stesso gene, o su larga scala producendo duplicazioni di interi clusters.

Per esempio, in

$$G_1 = (\ f\ \ e\ \ c\ \ \mathbf{e}\ \ \mathbf{b}\ \ \mathbf{e}\ \ d\ ) \text{ e } G_2 = (\ a\ \ \mathbf{b}\ \ \mathbf{b}\ \ \mathbf{e}\ \ c\ \ b\ \ c\ \ d\ \ \mathbf{b}\ \ \mathbf{e}\ \ c\ )$$

l'insieme {**b**, **e**} è un intervallo comune con una occorrenza in G_1, (e b e), e due occorrenze in G_2, (b b e) e (b e). Gli altri intervalli comuni sono {b}, {c}, {e}, {c, e}, {b, c, e} e {b, c, d, e}.

La differenza maggiore, ed anche il punto di forza del modello rispetto agli intervalli comuni su permutazioni, è che un intervallo comune sull'insieme di stringhe $\mathcal{G}$ può avere due o più occorrenze.

Dal momento che ogni occorrenza di un intervallo comune è senza *gaps*, più occorrenze dello stesso intervallo non si sovrappongono. Ma allora ciascuna sottostringa del genoma $\mathcal{G}$ può rappresentare un'occorrenza di al più un cluster di geni; e questo porta immediatamente alla seguente proposizione:

Proposizione 2.2. *Sia* n *la somma delle lunghezze delle stringhe in* $\mathcal{G}$*. Il numero e le occorrenze di intervalli comuni in* $\mathcal{G}$ *sono in* $\mathcal{O}(n^2)$*.*

Dunque, come nel caso di intervalli comuni su permutazioni, il massimo numero di intervalli comuni è quadratico rispetto alla dimensione del genoma. Questa proprietà deriva direttamente dall'assunto di non avere *gaps*. Un'importante differenza fra i due modelli consiste però nella struttura interna degli intervalli comuni su permutazioni (§ 2.4): l'esistenza di un *kernel* di intervalli irriducibili, di dimensione lineare, fornisce algoritmi lineari in spazio e tempo (§ 2.3.1). Attualmente, non è stata ancora scoperta un'analoga proprietà strutturale per gli intervalli su stringhe. Al successivo § 2.5.2.1 si analizzerà una proposta per trasformare stringhe con pochissimi duplicati in permutazioni.

La mancanza di una struttura interna agli intervalli comuni su stringhe fa sì che gli algoritmi usati siano nella natura molto diversi rispetto a quelli su

permutazioni, e molto meno eleganti: si basano infatti sull'enumerazione dei *fingerprints* [7] di $\mathcal{G}$.

Definizione 2.5. *Sia $S = s_1 \dots s_n$ una stringa definita sull'alfabeto Σ; il* fingerprint *di s è il sottoinsieme $\mathcal{F} \subseteq \Sigma$ che contiene quei caratteri di Σ che appaiono in S:*

$$\mathcal{F}(S) := \{s_i \mid 1 \leq i \leq n\} \subseteq \Sigma$$

Esempio 2.5. *Sia $g = CCGTCT$ una stringa definita sull'alfabeto $\Sigma = \{A, C, G, T\}$. Il* fingerprint *di g, $\mathcal{F}(g)$, è $\{C, G, T\}$.*

Segue immediatamente che un intervallo comune di $\mathcal{G}$ è uno dei *fingerprint* di $\mathcal{G}$ che appare, almeno una volta, in ogni stringa di $\mathcal{G}$. Dunque calcolare gli intervalli comuni su $\mathcal{G}$ si riduce ad un'appropriata ricerca su tutti i *fingerprints* di $\mathcal{G}$ con le loro relative posizioni sulle stringhe.

2.4 Notazione massimale

Il *pattern discovery* è uno dei principali metodi usati nello studio di sequenze genomiche. Quando si decide di modellare i cluster di geni imponendo il vincolo che siano intervalli comuni su permutazioni, la formulazione di *notazioni massimali*, senza introdurre perdita d'informazione, cattura importanti caratteristiche sulla struttura interna dei *motif* permutativi e ne riduce drasticamente il numero da analizzare, conferendo un senso all'enorme numero di risultati.

L'approccio tipico prevede infatti l'applicazione di uno specifico algoritmo che restituisce in uscita tutti i pattern trovati. Questo risultato, considerata la mole di dati trattati, ha l'inconveniente di rendere poco intellegibili le informazioni sull'ordinamento genico di ciascuna occorrenza dei numerosi pattern individuati.

Per questi motivi si è formalizzato il pattern discovery come un problema di pattern permutativi, o π–pattern, ricavandone una notazione massimale [13]. Siano $P = p_1, \dots, p_m$ e $S = s_1, s_2, \dots, s_n$ due sequenze di caratteri, con possibili ripetizioni, su un alfabeto Σ. Il pattern P appare alla locazione i in S *sse*

[7] Anche chiamato *character set* $\mathcal{CS}$.

$(p_1, p_2, \ldots, p_m)$ è una permutazione di $(s_i, s_{i+1}, \ldots, s_{i+m-1})$: P è un π*–pattern* se esiste un intero k per cui P appaia k volte in S.

Si noti che la definizione di π–pattern è l'equivalente in *information retrieval* della definizione 2.2 sugli intervalli comuni in permutazioni di geni.

Una notazione per i π–pattern massimali è stata introdotta come modello per filtrare delle permutazioni di caratteri rilevanti da altre apparentemente senza significato (v. esempi nel § 2.2). Un π–pattern p_1 è non–massimale rispetto al π–pattern p_2, se ogni occorrenza di p_1 è inclusa in un'occorrenza di p_2 e se ciascuna occorrenza di p_2 include un'occorrenza di p_1.

Definizione 2.6 (**Notazione massimale**)**.** *Date k permutazioni che rappresentano le k occorrenze di un pattern permutativo π, la notazione massimale è ottenuta interponendo:*

'–' *(lineetta) fra due gruppi di uno o più caratteri ad indicare che questi due gruppi appaiono sempre accostati in tutte le k permutazioni;*

',' *(virgola) in tutti gli altri casi.*

Esempio 2.6. *Si consideri il pattern $\{a, b, c, d, e, f\}$ che occorre una volta come 'abcdef' e una volta come 'bdacfe': la notazione massimale del pattern è $((a, b, c, d) - (e - f))$.*

La notazione massimale presenta due vantaggi. (1) Dà informazioni sulla struttura interna del pattern: $((a, b, c, d) - (e - f))$ mostra che e appare sempre affianco ad f ed entrambi sono sempre a fianco del gruppo $\{a, b, c, d\}$. (2) La notazione fornisce informazioni sulle relazioni non-massimali fra patterns e può essere utile per filtrare delle permutazioni di caratteri rilevanti da altre apparentemente senza significato (o non-massimali): $((a, b, c, d) - (e - f))$ mostra che i pattern $\pi_1 = \{e, f\}$ e $\pi_2 = \{a, b, c, d\}$ sono non-massimali rispetto al pattern $\pi_3 = \{a, b, c, d, e, f\}$ che li comprende entrambi. Dunque la notazione $((a, b, c, d) - (e - f))$ contiene tutta l'informazione delle occorrenze dei pattern π_1, π_2 e π_3.

2.5 La Genomica vista da un albero PQ

Un altro importante mezzo d'indagine è costituito dagli alberi PQ, che si sono rivelati degli ottimi strumenti per catturare e rappresentare le relazione permutative interne delle sequenze genomiche.

Dall'inizio dell'era genomica, sono state molte le tecniche e le nozioni presentate e, tra queste, gli alberi PQ sono stati usati a più riprese sia come strutture ausiliarie di verifica o dimostrazione [26, 12, 27, 28, 18], sia come elementi centrali di soluzioni innovative [14, 16].

Fatta eccezione per Karp [26], in cui si affrontano ancora i problemi della *mappatura* del filamento (1993), tutti gli altri contributi fanno riferimento alla comparazione genomica ad alto livello, ovvero considerando i geni piuttosto che i nucleotidi come la più piccola entità.

La comparazione genomica ad alto livello vede i geni, e le loro permutazioni, come il risultato di pressioni selettive e divergenze evolutive, responsabili, queste, degli aspetti funzionali dei geni. In questo contesto si ha un grosso alfabeto di geni, tutti distinti, con relativamente poche ripetizioni, le quali si presentano soprattutto in confronti eseguiti fra sequenze paraloghe.

2.5.1 Alberi PQ

Un albero PQ è una struttura dati ad albero, introdotta nel 1976 da K. Booth e G. Lueker [29], che rappresenta una famiglia di permutazioni su un insieme di elementi. L'albero rappresenta le permutazioni attraverso riordinamenti permissibili dei figli dei propri nodi.

L'albero PQ è un albero con radice e nodi interni etichettati, nel quale ogni elemento dell'insieme è una foglia e ogni nodo non-foglia può avere l'etichetta P o Q. I figli di un nodo P possono occorrere in un ordine arbitrario (dunque tutte le $k!$ permutazioni, se k è il numero dei figli), mentre i figli di un nodo Q occorrono in due sole disposizioni: nell'ordine originale oppure in ordine inverso.

Un albero PQ rappresenta tutti gli ordinamenti dei nodi foglia che è possibile descrivere permutando le foglie secondo una qualunque sequenza di queste due

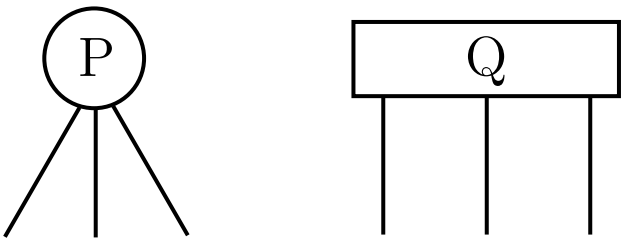


Figura 2.1: Rappresentazione grafica dei nodi P e Q.

disposizioni dei nodi P e Q. Si usa rappresentare un nodo di tipo P con un cerchio ed un nodo di tipo Q mediante un rettangolo (Figura 2.1).

La *frontiera* dell'albero T, indicata con $F(T)$, è quella permutazione indotta dall'albero T sulle foglie lette da sinistra a destra.

Definizione 2.7 (**Alberi PQ Equivalenti**)**.** *Due alberi PQ T e T' sono equivalenti, indicato $T \equiv T'$, se uno dei due può essere ottenuto dall'altro applicando una sequenza delle seguenti regole di trasformazione:*

1. *permutare arbitrariamente i figli di un nodo P;*
2. *invertire l'ordine dei figli di un nodo Q.*

La frontiera di un albero T' equivalente a T si dice *consistente* con T e si definisce l'insieme di tutte le frontiere consistenti con T come $\mathcal{C}(T) = \{F(T')|T' \equiv T\}$. Di conseguenza indichiamo il numero di frontiere ottenibili da un albero equivalente a T con $|\mathcal{C}(T)|$.

Chiaramente la relazione di equivalenza è riflessiva, simmetrica e transitiva, e quando $T \equiv T'$ allora risulta anche $\mathcal{C}(T) = \mathcal{C}(T')$.

Gli alberi PQ sono in *forma canonica* quando ogni nodo Q ha almeno due figli, e ogni nodo P almeno tre. Risulta immediato convertire ogni albero PQ nella propria forma canonica. Infine, esistono due alberi particolari: (1) sia N un insieme finito, l'albero costituito da un solo nodo P con $|N|$ figli, tutti nodi foglia, è chiamato *albero PQ universale* e si indica con T_U; (2) l'*albero nullo*, cioè

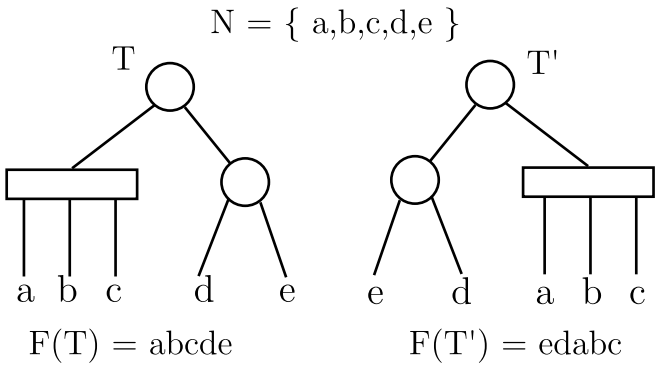


Figura 2.2: Due alberi PQ equivalenti, T e T'. Si noti che $\mathcal{C}(T) = \mathcal{C}(T') = \{abcde, abced, cbade, cbaed, deabc, decba, edabc, edcba\}$.

un albero che non ha nodi. Convenzionalmente non ha frontiera e il suo insieme di permutazioni consistenti è vuoto: $\mathcal{C}(T) = \emptyset$.

Booth e Leuker hanno realizzato gli alberi PQ per risolvere il *general consecutive arrangement problem*:

> *Dato un insieme finito N e una collezione $\mathcal{I}$ di sottoinsiemi di N, determinare se esiste una permutazione π di N per cui ogni sottoinsieme $i \in \mathcal{I}$ appare come una sottostringa consecutiva di π.*

Per risolvere questo problema hanno ideato un algoritmo che usa gli alberi PQ, di complessità $\mathcal{O}(n^2)$, che implementa la funzione $REDUCE()$ così definita:

Definizione 2.8 ($REDUCE(\mathcal{I}, T')$)**.** *Siano $\mathcal{I}$ una collezione di sottoinsiemi di $N = \{1, 2, \ldots, n\}$ e T' un albero PQ le cui foglie siano etichettate con $\{1, 2, \ldots, n\}$, la funzione $REDUCE(\mathcal{I}, T')$ costruisce un albero PQ T tale che $f \in \mathcal{C}(T)$ sse $f \in \mathcal{C}(T')$, ed inoltre ogni sottoinsieme $i \in \mathcal{I}$ appare come sottostringa consecutiva di f.*

La funzione $REDUCE(\mathcal{I}, T')$, che produce quindi alberi T equivalenti a T', restituisce l'albero nullo se non esiste una possibile frontiera $f \in \mathcal{C}(T')$ per cui ogni $i \in \mathcal{I}$ appare come sottostringa consecutiva di f.

Si noti che, se T_U è l'albero universale, allora l'applicazione della funzione $REDUCE(\mathcal{I}, T_U)$ non introduce alcun vincolo sulla creazione di alberi equivalenti ad un albero universale T_U e, quindi, semplicemente *restituisce un albero PQ T tale che $f \in \mathcal{C}(\mathcal{T})$ sse ogni sottoinsieme $i \in \mathcal{I}$ appare come sottostringa consecutiva di f.*

Questa è l'osservazione che sta alla base dell'applicazione degli alberi PQ alle sequenze genomiche.

2.5.1.1 Minimal consensus PQ tree

Dovendo tradurre *minimal consensus PQ tree* dall'inglese all'italiano, potremmo definirlo come "l'albero PQ minimale di massima inclusione".

Nel precedente § 2.4 abbiamo definito il concetto di notazione massimale per un insieme di permutazioni. Ora vogliamo definire il minimal consensus PQ tree [16] come uno strumento che ci permetta di ottenere, algoritmicamente, la notazione massimale delle k occorrenze di un π–pattern.

In analogia alla Definizione 2.6 (pag. 31), si prenda in considerazione l'insieme $\mathcal{C}(T)$ delle frontiere consistenti con un albero PQ:

Definizione 2.9 (**Notazione di un albero PQ**)**.** *La notazione di un albero PQ è ottenuta scrivendo la frontiera dell'albero come una stringa parentesizzata in cui la ',' (virgola) codifica i nodi P e la '–' (lineetta) in nodi Q.*

Esempio 2.7. *La notazione dell'albero T in Figura 2.2 (pag. 34) si indica con $((a - b - c), (d, e))$.*

La notazione massimale di un insieme di permutazioni Π risulta essere la stessa rappresentazione data per gli alberi PQ. A questo punto è dunque lecito mettere in relazione un insieme di permutazioni con un albero PQ T.

In particolare, si vorrebbe poter costruire un albero PQ T tale che $\mathcal{C}(T) = \{\pi_1, \pi_2, \ldots, \pi_k\} = \Pi$; purtroppo non è sempre possibile. Si consideri, infatti, il π–pattern $\{a, b, c, d, e\}$ che appaia quattro volte come in $\Pi = \{abcde,\ abced,\ cbade,\ edabc\}$; l'albero PQ T mostrato in Figura 2.2 (pag. 34), è quello che

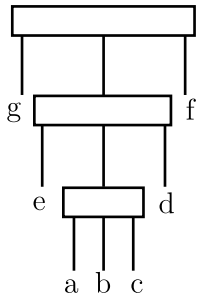


Figura 2.3: Sia π_1 = geabcdf e π_2 = fecbadg. L'albero PQ T in figura è il minimal consensus PQ tree di $\{\pi_1, \pi_2\}$. Si usa questo esempio per illustrare le tre differenti idee esposte nel testo.

meglio descrive tali occorrenze. Tuttavia il pattern *edcba* $\in \mathcal{C}(T)$ ma *edcba* $\notin \Pi$, ovvero viene generato dalla frontiera dell'albero ma non appare fra le reali occorrenze. D'altro canto, si noti anche che l'albero PQ universale T_U, è tale per cui $\{\pi_1, \pi_2, \ldots, \pi_k\} \subseteq \mathcal{C}(T_U)$. Da qui nasce l'idea e l'esigenza che esso sia anche *minimale*.

Definizione 2.10 (**Minimal consensus PQ tree**)**.** *Dato un insieme di permutazioni* $\Pi = \{\pi_1, \ldots, \pi_k\}$*, un* consensus PQ tree *T di Π è tale per cui $\Pi \subseteq \mathcal{C}(T)$. Il consensus PQ tree è minimale quando non esiste un $T' \not\equiv T$ tale che $\Pi \subseteq \mathcal{C}(T')$ e $|\mathcal{C}(T')| < |\mathcal{C}(T)|$.*

Sia Π l'insieme di k permutazioni $\pi_1, \pi_2, \ldots, \pi_k$ che rappresentano le k occorrenze del π–pattern, si dimostra [16] che:

> *Dato il π–pattern, è possibile costruire un minimal consensus PQ tree T la cui notazione è la notazione massimale del π–pattern, e ogni sottografo dell'albero PQ corrisponde ad una permutazione non-massimale del pattern.*

La Figura 2.3 illustra le proprietà dei minimal consensus PQ tree:

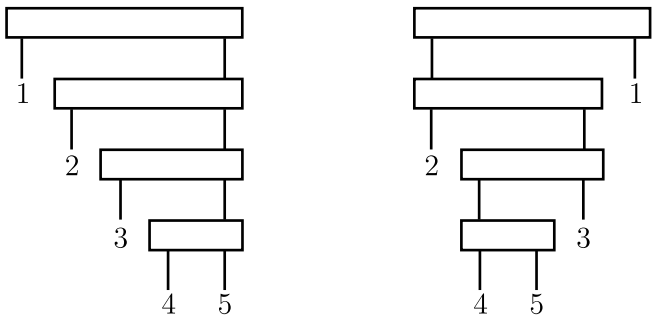


Figura 2.4: Siano $\pi_1 = 12345$ e $\pi_2 = 24531$ due stringhe di lunghezza 5. L'albero PQ (1 – (2 – (3 – (4 – 5)))) in figura è il minimal consensus PQ tree di $\{\pi_1, \pi_2\}$ e ha altezza 4.

Consensus Tree Se T_U è l'albero PQ universale, allora $\pi_1, \pi_2 \in \mathcal{C}(T) \subseteq \mathcal{C}(T_U)$ e dunque due stringhe possono non dar luogo ad un unico[8] consensus tree;

Potere espressivo Sia $\pi_3 =$ gdcbaef, chiaramente $\pi_3 \neq \pi_1$ e $\pi_3 \neq \pi_2$, e tuttavia $\pi_3 \in \mathcal{C}(T)$;

Altezza degli alberi PQ Un consensus PQ tree di due sole stringhe (permutazioni) di lunghezza L può arrivare ad un altezza di $\mathcal{O}(L)$ (v. Figura 2.4).

2.5.2 Analisi di prossimità fra geni

Di particolare interesse è il contributo fornito da Landau *et al.*[9] [16], nel quale si dà la dimostrazione che i minimal consensus PQ tree sono equivalenti con la notazione massimale dei π–pattern (insiemi di permutazioni). È proposto un miglioramento dell'algoritmo di Booth e Leuker, in grado di ottenere anche la notazione massimale di un π–pattern, che esibisce complessità lineare.

[8] Tuttavia, dato l'insieme di permutazioni Π, il *minimal consensus tree* è unico, a meno di equivalenza.

[9] Laxmi Priya Parida, coautore, il 31 luglio 2006, ha depositato la domanda di brevetto statunitense per l'invezione "Methods and systems for reconstructing common ancestors include determining a PQ tree structure based upon permutations between two genomes, and reconstructing an ancestor genome based upon the PQ tree structure. A PQ tree includes a first internal node (P node) that allows a permutation of the children thereof, and a second internal node (Q node) that maintains a unidirectional order of the children thereof".

Tale algoritmo, chiamato $REPLACE()$, costruisce sia un *minimal consensus PQ tree* e, quindi, anche la notazione massimale dell'insieme di permutazioni, ma anche una rappresentazione grafica del relativo albero PQ.

L'algoritmo $REPLACE()$, sfruttando le strutture dati usate in [8] per la costruzione degli intervalli irriducibili (Def. 2.3 pag. 28), esibisce complessità lineare. Tutti i simboli (i geni) delle sequenze su cui opera l'algoritmo per la costruzione degli alberi PQ sono distinti, ovvero le sequenze sono permutazione degli stessi n geni (v. § 2.3.1).

Il lavoro continua trovando cluster di geni permutati che, sotto forma di insiemi di permutazioni, vengono dati in input all'algoritmo. Per esempio, gli interi genomi dell'uomo e del ratto sono stati opportunamente pre-processati per creare un *mapping* biunivoco fra i due. Dopo aver enumerato, rispettivamente, con la permutazione identità (1, 2, ..., 25422) il genoma umano e una permutazione di questa il genoma del ratto, si è riusciti ad individuare geni funzionalmente correlati.

2.5.2.1 Gestire le molteplicità

Nel caso di sequenze biologiche, con geni che si presentano più di una volta, è necessario codificare le ripetizioni.

Si consideri il pattern p con le occorrenze *acbdefc* e *cdabfec*. Si nota come il simbolo c si ripete più volte nelle due stringhe. Considerando la molteplicità, si prenda *c'* come un nuovo simbolo. A questo punto le due occorrenze hanno tutti i simboli distinti e, dunque, p ha un unico minimal consensus PQ tree corrispondente alle due occorrenze *acbdefc'* e *cdabfec'*. Tuttavia, quando i simboli possono apparire più di una volta in un π–pattern, il minimal consensus PQ tree può non essere unico. Per illustrare questo concetto si fa riferimento alle Figure 2.5 e 2.6.

Si consideri il seguente insieme di permutazioni $\Pi = \{p_1, p_2, p_3\}$ con:
$p_1 = deabcxc$, $p_2 = cdeabxc$ e $p_3 = cxcbaed$.

Ogni carattere della sequenza di riferimento p_1 viene etichettato con un nume-

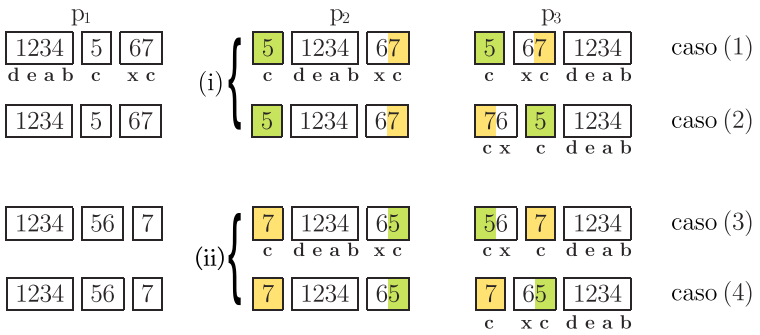


Figura 2.5: Siano p_1=*deabcxc*, p_2=*cdeabxc* e p_3=*cxcbaed*. Enumerando la prima stringa p_1=*deabcxc*=1234567, di conseguenza si ottiene p_2=*cdeabxc*=[57]12346[57] e p_3=*cxcbaed*=[57]6[57]4321. Eseguendo le due possibili scelte per i caratteri in p_2 si ha: (scelta **i**) p_2=$\underline{5}$123467, da cui (1) p_3=$\underline{5}6\underline{7}$4321 oppure (2) p_3=$\underline{76}\underline{5}$4321; (scelta **ii**) p_2=$\underline{7}$123465, da cui (3) p_3=$\underline{5}6\underline{7}$4321 oppure (4) p_3=$\underline{76}\underline{5}$4321.

ro distinto; le rimanenti sequenze non sono più viste come un insieme di caratteri, ma sono trattate piuttosto come *multiset*: stringhe i cui elementi costitutivi sono degli insiemi di caratteri. Nell'esempio si ha p_1 = *deabcxc* = 1234567 e, considerando la molteplicità del carattere *c*, segue p_2 = *cdeabxc* = [57]12346[57] e p_3 = *cxcbaed* = [57]6[57]4321. Modellando la molteplicità dei caratteri ripetuti con i multinsiemi, occorre poi esplicitare i diversi elementi di ogni insieme in un processo che prevede dunque tutte le disposizioni, senza ripetizione, di ciascun carattere in ogni insieme.

La Figura 2.5 mette in evidenza i quattro gruppi di *sequenze codificate* dal procedimento di sostituzione: si può osservare quali permutazioni subiscano i numeri della codifica, rispetto alla stabilità dei caratteri originari i quali, proprio in virtù delle molteplicità nelle occorrenze, possono essere scambiati senza alterare la sequenza.

In Figura 2.6 sono rappresentati gli alberi costruiti a partire dalle permutazioni ottenute dalla codifica. Le foglie con caratteri duplicati sono evidenziate secondo lo schema della codifica. Si noti, infine, che gli alberi T_1 e T_3, relativi ai casi (1) e (3), hanno la medesima rappresentazione.

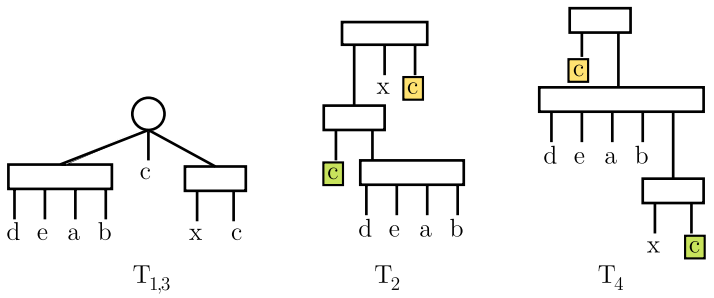


Figura 2.6: I tre alberi rappresentano i quattro casi mostrati in Figura 2.5: $T_{1,3}$ rappresenta i casi (1) e (3), T_2 e T_4 rappresentano il secondo e il quarto caso. Si noti che T_2 e T_4 sono entrambi minimal consensus PQ tree di $\{p_1, p_2, p_3\}$ dal momento che $|\mathcal{C}(T_2)| = |\mathcal{C}(T_4)| = 8 < |\mathcal{C}(T_{1,3})|$.

2.5.3 Analisi di sequenze nucleotidiche

Recentemente è stata applicata la metodologia degli alberi PQ anche alle sequenze nucleotidiche [30], ovvero tornando a considerare i nucleotidi come la più piccola entità permutativa.

Nella sua memoria di laurea, Camposeo spiega in che modo, dopo gli iniziali e abbondanti risultati di "uno studio delle permutazioni contenute all'interno delle sequenze analizzate, è stato fatto ricorso ad un metodo per poter filtrare le permutazioni più interessanti mediante l'introduzione e l'utilizzo della struttura dati degli alberi PQ".

La metodologia ha "messo in evidenza alcune zone delle sequenze genomiche analizzate, che presentavano particolari proprietà permutative, sulle quali bisognerà effettuare uno studio biologico più approfondito per vedere se [...] hanno anche determinate proprietà biologiche".

L'applicazione dello schema generale riportato in Landau *et al.*[16] a questo ambito richiedeva due accortezze particolari:

1. Una codifica delle sequenze, che ora sono milioni di ripetizioni di soli quattro simboli: le basi azotate {A, C, G, T};

2. L'identificazione, sulle sequenze così codificate, dei cluster di permutazioni da analizzare con l'algoritmo.

2.5.3.1 Codifica delle sequenze

In questo caso il problema è codificare una stringa altamente ridondante in un'altra stringa, della stessa lunghezza, ma usando tanti simboli quanti sono tutti i caratteri che la compongono.

La gestione della molteplicità proposta in Landau *et al.* è molto interessante e facilmente applicabile nel caso di pochissime ripetizioni rispetto all'alfabeto di base. Ricordiamo che quello schema (v. § 2.5.2.1) prevede essenzialmente di enumerare la prima stringa con la permutazione identità $\pi_{id} = \{1, 2, \ldots, n\}$, assegnando dunque un nuovo simbolo ad ogni successivo carattere, unico o ripetuto che sia. Le rimanenti stringhe vengono invece trattate come multiset: ovvero ogni carattere che si ripete, viene sostituito dall'insieme di tutti i nuovi simboli unici che codificano le ripetizioni nella prima stringa.

Con questo tipo di codifica è però necessario esplicitare, senza introdurre ripetizioni, tutti gli insiemi che sostituiscono le ripetizioni di caratteri. È facile intuire che questa operazione produce un numero di permutazioni dello stesso ordine di grandezza del numero di disposizioni senza ripetizioni delle quattro basi {A, C, G, T}:

$$\frac{n!}{n_a! \cdot n_c! \cdot n_g! \cdot n_t!}$$

Dove n è il numero totale di caratteri, mentre gli n indiciati rappresentano il corrispondente numero di basi.

Nel caso trattato da Camposeo, riducendo la complessità esponenziale del problema, è stato deciso di codificare le stringhe senza ricorrere agli insiemi:

1. si enumera la prima stringa con la permutazione identità $\pi_{id} = \{1, 2, \ldots, n\}$;

2. si scorre la seconda stringa, assegnando ad ogni nuovo carattere incontrato il numero associato alla prima occorrenza dello stesso carattere nella prima stringa.

Figura 2.7: Nell'esempio, con n=6, si ha S_1 = "atcaat" = $\pi_2(S_2)$ = "ttacaa" = $\pi_3(S_3)$ = "caatat".

Esempio 2.8. *Le due sequenze ACACACA e CACACAA vengono codificate come segue:*

A	C	A	C	A	C	A		C	A	C	A	C	A	A
1	2	3	4	5	6	7		2	1	4	3	6	5	7

Tabella 2.1: La codifica proposta da Camposeo produce, in questo esempio, un'inversione a coppie degli elementi.

Si noti che, per applicare questo tipo di codifica, le due permutazioni, oltre ad avere lo stesso insieme di caratteri, devono avere la stessa lunghezza.

2.5.3.2 Ricerca delle permutazioni

Per quanto riguarda l'identificazione delle permutazioni, Camposeo ha fatto scorrere una finestra, di dimensione n=200, lungo l'intera sequenza, cercando, ad ogni avanzamento, tutte le sottostringhe, permutazioni della finestra corrente, nel resto della sequenza biologica.

L'avanzamento della finestra equivale a scegliere una nuova sottosequenza S_i, di dimensione 200, da usare come sequenza di riferimento nella ricerca di permutazioni. Siano, in numero, $p(i)$ tutte le permutazioni di S_i trovate, allora la stringa S_i determina l'insieme $\Pi_i = \{S_k | S_k = \pi(S_i), 1 \leq k \leq p(i)\}$.

Su sequenze di 400 mila basi azotate, quel procedimento ha prodotto oltre 300 mila permutazioni, ciascuna di lunghezza 200, ripartite in più insiemi Π_i, con $|\Pi_i| = p(i) \approx 15$ e $1 \leq i \lesssim 25000$.

Questa operazione produce essenzialmente una ripartizione in classi di equivalenza[10] Π_i le quali contengono mediamente 15 elementi. Ad ognuna di queste classi Π_i è stato applicato l'algoritmo di Landau $REPLACE()$ e creata dunque la notazione massimale del π–pattern Π_i.

In quella sede si è notato che molte delle stringhe permutate, poi usate nella costruzione degli alberi, presentavano, comprensibilmente, sovrapposizione reciproche a causa delle piccole dimensioni dell'alfabeto {A,C,G,T} e dell'enorme lunghezza della sequenza.

2.6 Strumenti per specialisti

Gli esperimenti biologici vengono spesso effettuati prima *in silico*, con l'utilizzo degli strumenti informatici, e successivamente *in vitro* per effettuare le verifiche di laboratorio e ulteriori test.

In questa sezione facciamo un rapido excursus sugli strumenti applicativi usati dai genetisti. Infatti molte delle questioni descritte nelle sezioni precedenti trovano risposte nell'uso degli strumenti attualmente disponibili. Tuttavia tali strumenti, proprio perché sviluppati direttamente sul campo, sono usati estensivamente sui vasti genomi e devono fornire risposte in tempi accettabili usando quindi delle euristiche.

2.6.1 Formato FASTA

FASTA è un formato testuale per rappresentare sequenze di acidi nucleici (o peptidiche), nel quale ogni base azotata (o aminoacido) è rappresentata da una lettera dell'alfabeto. È anche possibile inserire commenti o altre informazioni all'inizio della sequenza.

[10] È immediato riconoscere le proprietà riflessiva, simmetrica e transitiva nella relazione di equivalenza indotta dalla funzione permutazione $\pi(\cdot)$.

Per esempio, tutti i frammenti di sequenze nucleiche in questa tesi, usando le lettere {A, C, G, T}, sono in formato FASTA. In Appendice B sono riportate le tabelle complete con la leggenda dei simboli.

La semplicità del formato FASTA permette l'implementazione di semplici algoritmi di manipolazione e parsing in linguaggi quali Perl o Python.

2.6.2 BLAST

BLAST[11] (Basic Local Alignment Search Tool) è uno strumento di ricerca di allineamenti locale. È un algoritmo usato per comparare le informazioni contenute nelle strutture biologiche primarie[12], come ad esempio le sequenze proteiche o le sequenze nucleotidiche delle molecole di DNA (v. cap.1). Una ricerca BLAST permette al ricercatore di confrontare una sequenze di interesse con un database di sequenze già conosciute, e di identificare tra queste ultime quelle che presentano delle somiglianze con la sequenza di interesse. Ad esempio, in seguito alla scoperta di un gene di topo, prima sconosciuto, gli scienziati tipicamente compiono una ricerca BLAST nel genoma umano per vedere se contiene geni con sequenze simili. BLAST trova le sequenze nel genoma umano che somigliano al gene del topo e su questi geni affini si può indagare sperimentalmente la funzione di quello proveniente dal topo per ottenere così indizi sulla possibile funzione di quello umano.

BLAST è specificamente indirizzato ad un fondamentale problema genomico e per questo è fra i programmi più usati in bioinformatica. Inoltre il suo algoritmo risulta essere particolarmente veloce anche sugli enormi genomi attualmente disponibili a scapito di risposte troppo raffinate, dal momento che deve necessariamente usare delle euristiche di ricerca che approssimano i lenti approcci esaustivi. L'idea algoritmica di base è quella di individuare le coppie di segmenti più rilevanti[13] contenuti in un allineamento statisticamente significativo [31]. BLAST

[11] http://blast.ncbi.nlm.nih.gov/.

[12] La struttura primaria di una biomolecola è la descrizione esatta della sua composizione atomica e dei legami presenti tra gli atomi.

[13] Noti come *high-scoring segment pairs* (HSP).

```
>ref|NT_025741.14|Hs6_25897 Download subject sequence spanning the
 HSP Homo sapiens chromosome 6 genomic contig, reference assembly
 Length=61645385

 Features flanking this part of subject sequence:
   69196 bp at 5' side: SNF2 histone linker PHD RING helicase isoform a
   3401 bp at 3' side: glutamate receptor, metabotropic 1

 Score = 2795 bits (1513),  Expect = 0.0
 Identities = 1575/1600 (98%), Gaps = 23/1600 (1%)
 Strand=Plus/Plus

Query  1         GACTTGCATGAGGCCTATGGCCTCTTTGTTCTGGGCAACTTCTTCCATTTGGAATGGGAA  60
                 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  50450083  GACTTGCATGAGGCCTATGGCCTCTTTGTTCTGGGCAACTTCTTCCATTTGGAATGGGAA  50450142

Query  61        TTTGCTAGTT---A---A--TC-------C------CC--TTCAATCCTACTGGAAGAAC  120
                 ||||||||||   |   |  ||       |      ||  ||||||||||||||||||||
Sbjct  50450143  TTTGCTAGTTTGTATGTAGCTCATTGTTGCAGTAATCCCCTTCAATCCTACTGGAAGAAC  50450202
```

Figura 2.8: In questo caso ho effettuato una ricerca di un segmento di 1600 basi del DNA umano, di cui si mostrano qui solo le prime 120. Ho preventivamente alterato due basi e rimosse altre 23. BLAST restituisce un risultato, con il 98% di confidenza, alla posizione 50450083 nel cromosoma 6 della specie di *Homo sapiens*.

viene dunque usato per inferire relazioni funzionali ed evolutive tra sequenze e così anche come ausilio nell'identificazione dei membri di famiglie di geni.

2.6.3 ClustalW

ClustalW[14] è un programma per l'allineamento multiplo di sequenze[15] divergenti di DNA o di proteine. Il programma calcola il miglior match delle sequenze e le allinea in modo tale da mettere in evidenza identità, somiglianze e differenze. Si rappresentano i risultati di un allineamento multiplo di sequenze con una *sequenza di consenso* in cui, dal confronto di sequenza correlate, vengono individuati *motif* di sequenze funzionalmente simili.

Solitamente, le sequenze date in input hanno delle relazioni evolutive per cui condividono una discendenza oppure derivano da un antenato comune. Dall'al-

[14] http://www.ebi.ac.uk/clustalw/.

[15] Tecnica nota come *multiple sequence alignment* (MSC).

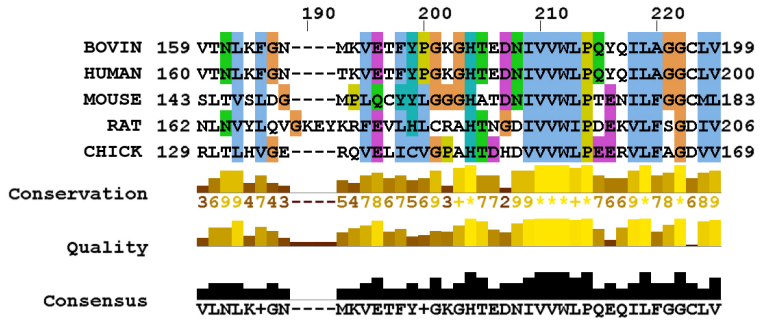


Figura 2.9: In questo caso ho eseguito l'allineamento fra le sequenze di proteine indicate. Nella parte superiore della figura sono evidenziate le somiglianze fra le sequenze, in quella inferiore la conservazione, la qualità e il consenso fra gli aminoacidi allineati.

lineamento ottenuto è possibile inferire omologie fra le sequenze o fare analisi filogeniche (per lo studio delle specie).

L'allineamento multiplo di sequenze evidenzia quali segmenti, o cluster di geni (cfr. 2.2), si preservano sempre uguali e quali sono variabili.

2.6.4 Primer3

Un *primer* è un filamento di acido nucleico che serve come punto di innesco per la replicazione del DNA.

I primer sono necessari perché molti degli agenti[16] per la replicazione del DNA non possono iniziare la sintesi di un nuovo filamento "ex novo", ma possono solo aggiungere nucleotidi ad un filamento pre-esistente.

I primer sono utilizzati in molte tecniche di biologia moleolare che richiedono la replicazione del DNA, come in alcune tecniche di sequenziamento del DNA e nella reazione a catena della polimerasi (PCR). I primer usati in queste tecniche sono dei corti frammenti di DNA, della lunghezza di 18–20 basi. Questi inneschi sono costruiti in laboratorio, attraverso una sintesi chimica.

Primer3 è il programma[17] più usato per modellare e analizzare i primer.

[16] DNA-polimerasi, enzimi che catalizzano la replicazione del DNA.
[17] http://primer3.sourceforge.net/

Capitolo 3

Un approccio a intervalli

Nel capitolo 2 abbiamo visto la dicotomia esistente fra la ricerca genetica sul campo da un lato, che quotidianamente ricava nuova conoscenza usando applicativi software specifici e ormai maturi, e la ricerca accademica dall'altro, impegnata a formalizzare i concetti biologici. Sebbene le applicazioni in uso risolvano già molti più problemi di quanti non ne siano stati formalizzati in accademia e, nonostante i vari risultati pubblicati vengano di rado utilizzati estensivamente dai genetisti, tuttavia è già constatabile l'uso di tecniche e algoritmi, spesso approssimati, provenienti da aree disciplinari diverse.

Quegli strumenti, proprio perché sviluppati direttamente sul campo, sono usati estensivamente su vasti genomi e forniscono risposte adeguate, e in tempi accettabili, usando delle euristiche.

Lo sforzo collettivo di definire dei modelli formali a fondamento della giovane scienza genomica è quindi mirato ad arricchire e rafforzare gli algoritmi pratici esistenti, i quali, assodata la vastità dei dati trattati, attualmente enfatizzano la velocità a scapito di risposte troppo raffinate.

Le numerose proposte segnalate e l'intensa attività della comunità su questo argomento testimoniano anche la fondamentale mancanza di un modello dell'informazione che, mutuando l'espressione dalla logica matematica, sia *corretto e completo*.

Nel § 2.5.3 abbiamo visto un approccio all'analisi di sequenze nucleotidiche, mutuato da un precedente algoritmo su sequenze di geni di Landau *et al.* analizzato al § 2.5.2.

Partendo dall'idea di usare gli alberi PQ per dedurre informazioni dalla sequenza nucleotidica, si propone qui una soluzione diversa, che prende anche spunto da osservazioni fatte sul modello di Camposeo, modello che indubbiamente costituisce il primo importante banco di prova per questo tipo di metodologia.

In questo contesto, presentiamo prima un approccio basato sulla ricerca di intervalli comuni con ripetizioni. Poi, al capitolo 4, proponiamo una metodologia ad intervalli mirata all'analisi delle sequenze altamente ridondanti di nucleotidi, utilizzando i Suffix Tree per effettuare la codifica e gli Alberi PQ per ottenerne la notazione massimale. Sono prima introdotte le motivazioni ad entrambe le proposte.

3.1 Motivazioni

Nella formulazione delle due soluzione proposte è possibile rintracciare parecchie idee, alcune delle quali si basano su osservazioni compiute su lavori precedenti, i quali hanno fornito ottimi spunti di riflessione. Rapidamente, riassumiamo lo stato dell'arte sulla codifica nucleotidica già spiegato al § 2.5.3.1, per poi continuare con le riflessioni che ispirano questa tesi.

Codifica nucleotidica

La gestione delle ripetizioni suggerita da Landau *et al.* [16] avrebbe prodotto un numero fattoriale di permutazioni, pari a $\frac{n!}{n_a!\cdot n_c!\cdot n_g!\cdot n_t!}$ in cui n è il numero totale di caratteri, mentre gli n indiciati rappresentano il corrispondente numero di basi.

Dovendo evitare la complessità esponenziale del problema, un primo approccio consiste nel codificare le stringhe senza ricorrere all'espansione dei multinsiemi: 1) si enumera la prima stringa con la permutazione identità $\pi_{id} = \{1, 2, \ldots, n\}$ e 2) si scorre la seconda stringa, assegnando ad ogni nuovo carattere incontrato il numero associato alla prima occorrenza dello stesso carattere nella prima stringa.

Vediamo, anche con degli esempi, quali sono i problemi che derivano dall'alta ridondanza, ma anche quali proprietà vorremmo preservare nell'individuazione dei pattern che rappresentano basi nucleiche.

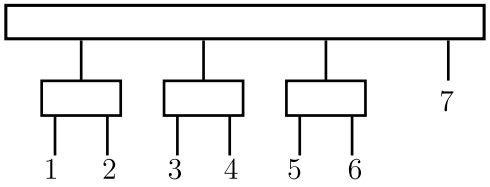


Figura 3.1: L'albero PQ costruito dall'algoritmo di Landau *et al.* associato con la notazione $((1-2)-(3-4)-(5-6)-7)$.

3.1.1 Alterazioni

Prendiamo in considerazione le due sequenze ACACACA e CACACAA. Presentano vari pattern interni e ci interessa conoscere quale sia una buona codifica per le ripetizioni. Adottando lo schema classico, vengono codificate come:

A	C	A	C	A	C	A
1	2	3	4	5	6	7

e

C	A	C	A	C	A	A
2	1	4	3	6	5	7

Si può notare un'inversione a coppie nella codifica numerica. Proseguendo con l'applicazione dell'algoritmo di Landau otteniamo la notazione massimale $((1-2)-(3-4)-(5-6)-7)$ che corrisponde all'abero PQ mostrato in Figura 3.1. La coppia di permutazioni e l'albero PQ associato sono una delle possibilità, fra tutte quelle ottenibili dall'applicazione della tecnica di espansione dei multinsiemi di Landau.

Questo tipo di codifica presenta lo stesso comportamento in tutti i casi di pattern strutturati a morfologia ridondante, ovvero pattern con sottostrutture interne ripetute. Per rendersene conto basta pensare alla dimostrazione del passo induttivo di una semplice induzione strutturale sui "blocchetti" interni (in questo caso "CA") del pattern.

In un caso in cui salta subito all'occhio la presenza di un pattern molto lungo,

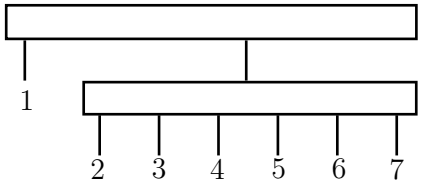


Figura 3.2: Il *minimal consensus PQ Tree* relativo alle stringhe ACACACA e CACACAA.

la domanda è se può essere preferibile una codifica che mantenga unito l'intero segmento:

A	C	A	C	A	C	A
1	2	3	4	5	6	7

e

C	A	C	A	C	A	A
2	3	4	5	6	7	1

Per rispondere a questa domanda serve il parere del genetista. Però possiamo generalizzare la questione e chiederci:

> *è possibile preservare, nella codifica, le sottostringhe comuni di lunghezza massima o comunque di una lunghezza minima fissata, piuttosto di trovare quelle di lunghezza minima?*

Si noti infine che il metodo descritto in Landau *et al.* definisce sempre un *minimal consesus PQ Tree* (v. § 2.5.1.1). Nel caso di ripetizioni, l'albero minimale va ricercato fra tutti quelli prodotti dall'espansione dei multinsiemi (§ 2.5.2.1). Se si codificano le due stringhe biologiche in maniera tale da mantenere compatta la sottostringa comune di massima lunghezza, allora la relativa notazione massimale $(1-(2-3-4-5-6-7))$ è esattamente quella generata dal *minimal* consesus PQ Tree. La notazione massimale calcolata con gli alberi PQ può essere scritta in termini delle due sequenze originali come $(A-(CACACA))$ e rappresentata come in Figura 3.2.

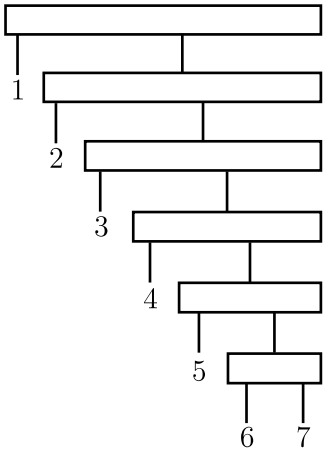


Figura 3.3: L'albero PQ di profondità 6 costruito dall'algoritmo di Landau *et al.* sulle codifiche delle stringhe AGCTGTT e GTTTGCA usando gli approcci esistenti.

3.1.2 Artefatti

Prendiamo in considerazione le due sequenze AGCTGTT e GTTTGCA: si possono già intravedere alcuni pattern al loro interno. Essendoci le ripetizioni occorre stabilire come enumerare le due stringhe. Tentiamo prima i metodi esistenti, ottenendo le permutazioni:

$$\begin{array}{ccccccc} A & G & C & T & G & T & T \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

e

$$\begin{array}{ccccccc} G & T & T & T & G & C & A \\ 2 & 4 & 6 & 7 & 5 & 3 & 1 \end{array}$$

Applicando l'algoritmo di Landau per la scoperta della notazione massimale si ottiene $(1-(2-(3-(4-(5-(6-7))))))$, che corrisponde all'albero di profondità 6 mostrato in Figura 3.3. Questo è anche l'albero di massima profondità ottenibile da due stringhe di quella lunghezza.

Guardando le due stringhe AGCTGTT e GTTTGCA vorremmo avere un sistema di codificarle numericamente in maniera tale da ottenerne il π–pattern

$(GTT - T - GC - A)$ ovvero in modo da assegnare la seguente numerazione:

$$\begin{array}{ccccccc} A & G & C & T & G & T & T \\ 1 & \underline{2} & \underline{3} & 4 & \underline{\underline{5}} & \underline{\underline{6}} & \underline{\underline{7}} \end{array}$$

e

$$\begin{array}{ccccccc} G & T & T & T & G & C & A \\ \underline{\underline{5}} & \underline{\underline{6}} & \underline{\underline{7}} & 4 & \underline{2} & \underline{3} & 1 \end{array}$$

In Figura 3.4 sono visibili tutti gli alberi risultanti dall'espansione dei multinsiemi di Landau (cfr. 2.5.2.1). L'albero con la cornice azzura è l'abero di massima altezza possibile per due stringhe lunghe 7. L'unico *minimal consensus PQ Tree*, che produce dunque il minor numero di frontiere, è l'albero nella cornice verde, ovvero l'albero risultante dalla codifica "intuitiva" che tiene insieme i segmenti GTT e GC.

3.1.3 Località e corrispondenze perfette

Dal momento che la gestione delle ripetizione è ancora uno dei problemi aperti, in letteratura le sequenze genomiche da codificare sono scelte, o preprocessate, in modo tale che siano già delle permutazioni con lo stesso numero di occorrenze per ciascun carattere.

Per esempio, in precedenti tentativi su sequenze nucleotidiche (v. § 2.5.3.2) si è fatto scorrere una finestra di dimensione $n = 200$ determinando una ripartizione in classi di permutazioni. All'interno di ciascuna classe vengono cercati i pattern che verranno poi processati per inferirne la notazione massimale, permettendo di trovare solo pattern di lunghezza massima pari alla dimensione della finestra. I pattern individuati si riferiscono agli elementi della classe di equivalenza, i quali si localizzano su specifici tratti della sequenza. In altri casi [22, 21], questa volta nella ricerca genomica ad alto livello, al fine di ottenere permutazioni senza ripetizioni, vengono rimossi a caso i geni duplicati. Ci chiediamo dunque

> *se sia possibile, e in che misura, operare analisi di tipo permutativo su sequenze di lunghezze diverse?*

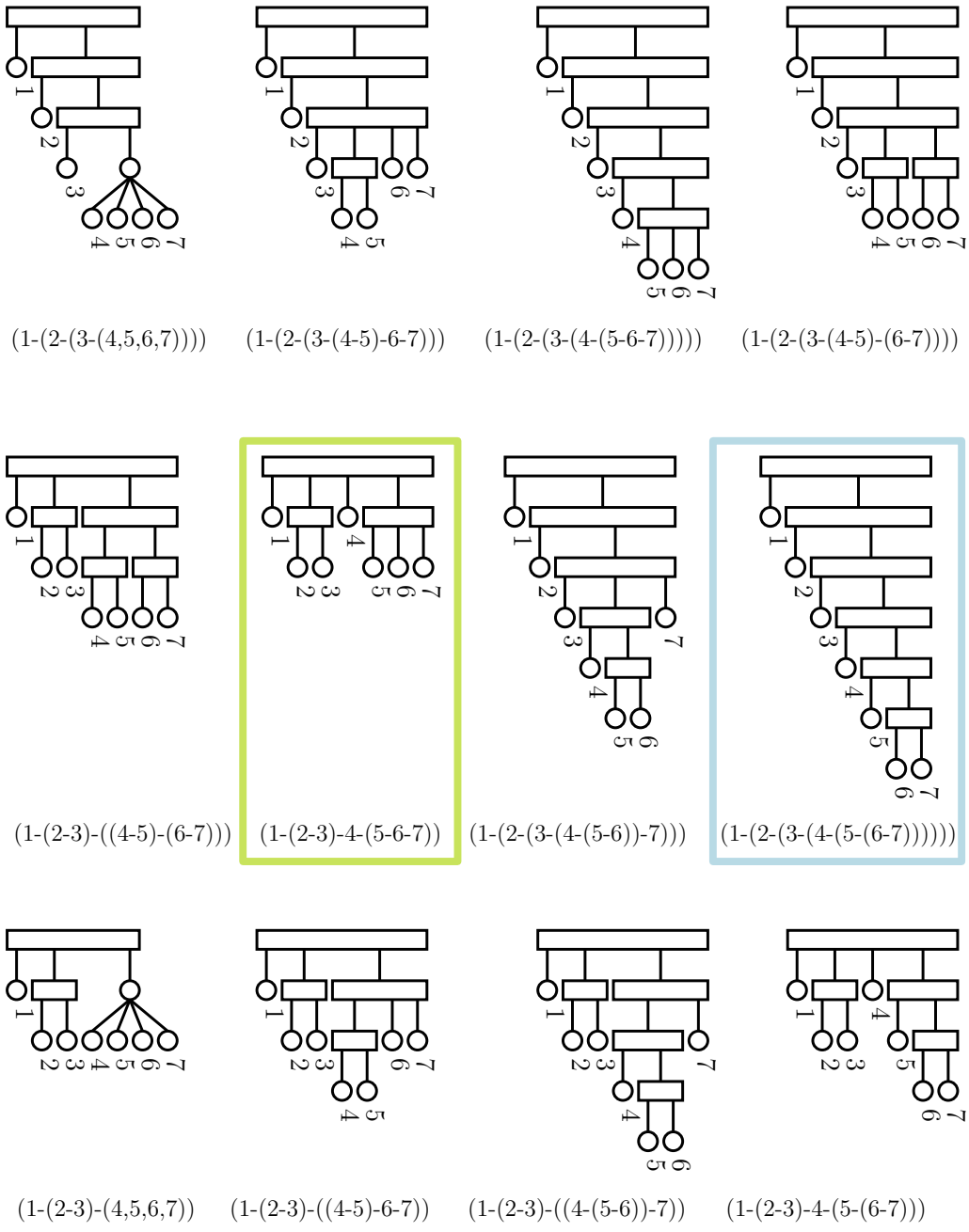


Figura 3.4: Tutti gli alberi sviluppati dai multinsiemi di Landau. Quello in verde è l'unico *minimal consensus PQ Tree*.

3.1.4 Struttura e significato delle ripetizioni

Formalizzare le ripetizioni con il modello dei multinsiemi (v. § 2.5.2.1) presuppone implicitamente una proprietà *ordinale* dei caratteri che si ripetono. Ovvero, la specifica posizione dei caratteri, singoli o con molteplicità maggiore di 1, diventa parte del modello nella formalizzazione delle ripetizioni. Sotto la pressione di produrre permutazioni *senza ripetizioni* totalmente ordinate, questa metodologia può risultare utile qualora ci siano poche ripetizioni e non sia disponibile, come nel nostro caso, un modello dei dati e, dunque, ci si ritrova a non potere, o volere, escludere nessuna possibile disposizione degli elementi.

Tuttavia, si può anche osservare che non è tanto importante l'*ordine* con cui quei caratteri si ripetono, quanto piuttosto il *contesto* in cui si ripetono. Enumerando con degli *ordinali* ciascuna ripetizioni, le possibili disposizioni di tali caratteri ripetuti, pur lasciando inalterata la sequenza genomica biologica, producono permutazioni numeriche in quantità fattoriale.

Ciò che manca è l'informazione *posizionale* dei caratteri con molteplicità: il contesto in cui si presentano porta con se quell'informazione biologica (v. cap. 1) che ancora non conosciamo ma su cui si concentra lo sforzo della genomica.

3.2 Esperimenti con intervalli comuni

Molte osservazione portano inevitabilmente a voler guardare i cluster di nucleotidi, non solo come permutazioni, ma anche come stringhe con ripetizioni. In letteratura sono state fatte varie proposte, già presentate e sistematizzate nel § 2.3 a cui si rimanda per i dettagli.

In fase di ricerca, indagando sulle proprietà peculiari delle sequenze con un numero di ripetizioni proporzionale all'intera lunghezza, ho condotto degli esperimenti sui *fingerprint*. Ricordiamo che il *fingerprint* (o *character set*) di una stringa s è il sottoinsieme dei caratteri dell'alfabeto che compongono s.

Ogni *fingerprint* individua sulle sequenze biologiche uno o più intervalli massimali, ovvero intervalli che sono delimitati a destra e a sinistra da un carattere non appartente al *fingerprint*, oppure l'estremità della sequenza.

Lunghezza	Numero	Locazioni
1	206327	8[t], 15[t] ...
2	112161	2[ac], **42**[t], ...
⋮	⋮	⋮
36	62	**100**[t], ...
⋮	⋮	⋮
187	1	500[ct]
217	1	430[cgt]
TOTALE	545234	

Tabella 3.1: Intervalli individuati sulla sequenza umana dai *fingerprint* comuni.

Si è usato l'algoritmo noto più efficiente [15] che gira in $\mathcal{O}(n^2)$. Sulla base dell'articolo pubblicato, si è implementato[1] *from scratch* in C# l'algoritmo che calcola gli intervalli comuni su stringhe. Sono state fatte delle scelte implementative che hanno reso utilizzabile l'output estremamente ridondante dell'algoritmo originale. Inoltre si sono apportate alcune ottimizzazione alle strutture dati interne sfruttando le proprietà derivanti dall'avere un alfabeto di quattro elementi.

Come ci si aspetta, dato l'alfabeto $\Sigma = \{A, C, G, T\}$, l'insieme dei *fingerprint* è un sottoinsieme dell'insieme delle parti $\mathcal{P}(\Sigma)$ con cardinalità[2] $|\mathcal{P}(\Sigma)| = 2^{|\Sigma|} = 16$.

Si è applicato l'algoritmo a due sequenze ciascuna di circa 400 mila basi, ma non della stessa lunghezza, appartenenti una all'uomo e l'altra al topo. Vediamo un estratto dei risultati.

In Tabella 3.1 sono riportati gli intervalli sulla sequenza umana determinati dai *fingerprint* comuni alla sequenza topesca. La colonna *Lunghezza* indica la lunghezza dell'intervallo individuato dal *fingerprint*, *Numero* indica il numero di intervalli di quella lunghezza e *Locazioni* riporta le posizioni alle quali appaiono gli intervalli con il relativo *fingerprint* fra parentesi [ ]. Sono in ordine crescente rispetto alla lunghezza degli intervalli e sono state cambiate alcune locazioni per renderle "leggibili".

[1] Il codice è disponibile su richiesta.

[2] L'insieme vuoto $\emptyset \in \mathcal{P}(\Sigma)$ corrisponde al *fingerprint* della stringa vuota.

fingerprint	Quantità
[a]	78842
[c]	58907
[g]	60279
[t]	82146
[t c]	30239
[t a]	**34398**
[g t]	31273
[g c]	**15666**
[a c]	29188
[g a]	29591
[g t c]	18181
[g t a]	22766
[g a c]	17280
[t a c]	22344
[g t a c]	**21**
TOTALE	531121

Tabella 3.2: Numero di intervalli determinati nella sequenza topesca dai *fingerprint* nucleoditici.

Leggendo le prime due righe, si vede che sono stati individuati oltre 200 mila singoletti, ovvero caratteri non affiancati da una loro ripetizione, e più di 100 mila coppie. Alla posizione 42 della sequenza ci sono due T, mentre alla posizione 100 è presente una serie di 36 T che, formando legami deboli con la A, indicano uno dei punti facilmente separabili nella doppia elica.

Infine notiamo le ultime due righe: sono due intervalli di lunghezza diversa, uno è incluso nell'altro: questo è possibile (cfr. 2.3.2) solo se i loro *fingerprint* sono diversi, e difatti $\{C, T\} \subset \{C, G, T\}$. Le innumerevoli inclusioni di *fingerprint* sono confermate dal totale di intervalli individuati, notabilmente maggiore della lunghezza della sequenza stessa (tuttavia il numero di singoletti copre oltre un terzo del totale).

Nella tabella 3.2, relativa alla sequenza topesca, si può notare a prima vista la differenza fra il numero di intervalli più "deboli", costituiti da sole {A, T}, dagli intervalli più "forti" con sole {C, G}. All'ultima riga, sono visibili 21 intervalli indotti dal *fingerprint* {A, C, G, T}, ossia da tutte e quattro le lettere dell'alfabeto. Questo si spiega facilmente andando a controllare la sequenza, nella quale

si rinvengono molti segmenti contenenti la lettera 'n' che, nel formato FASTA (§ 2.6.1 e app. B), significa 'aNy', ovvero ci sono almeno 20 punti in cui appare una base non determinata con precisione.

Moltissime strutture combinatorie, alcune delle quali davvero affascinati, sono state individuate e memorizzate[3]. Per ogni *fingerprint* sono disponibili svariate viste di tabelle o statistiche, sia essa la sequenza umana o topesca; tuttavia solo il responso di un biologo può stabilire quali intervalli sono davvero interessanti.

Data l'altissima ridondanza, risulta necessario analizzare più dettagliatamente questi stessi risultati utilizzando metodi automatici secondo le indicazioni suggerite dagli specialisti. Tuttavia preme sottolineare che queste analisi non impongono un modello, permutativo o qualt'altro, ma evidenziano la presenza di sottoinsiemi in comune fra le sequenze. Tali sottoinsiemi potranno dunque essere soggetti a successiva modellazione per il discovery di pattern interessanti.

[3]Da 800KB di dati in input sono stati generati 53MB di risultati testuali.

Capitolo 4

Una proposta a segmenti

Fra le motivazioni al § 3.1 abbiamo visto che alcune sequenze nucleotidiche, rappresentate come insiemi di permutazioni, si modellano più efficacemente quando viene preservata la contiguità delle sottostringe comuni ripetute. Infatti (cfr. 2.5.2), nei segmenti comuni a due sequenze, eventuali elementi duplicati devono essere modellati come multinsiemi per poi estrarne tutte le possibili disposizioni senza ripetizioni. Dall'espansione dei multinsiemi si ottengono dunque delle famiglie di permutazioni, in ciascuna delle quali si cercano gli intervalli irriducibili che saranno poi passati in input all'algoritmo $REPLACE()$. Per ogni famiglia, l'algoritmo genera la notazione massimale relativa ad un *consensus PQ Tree*, trasformando sequenze contigue in nodi Q. Fra tutti gli alberi generati si sceglie l'albero T che genera il minor numero di frontiere, il *minimal consensus PQ Tree*. Si osservi che il numero di frontiere $|\mathcal{C}(T)|$ ottenibili da T è tanto più grande quanto maggiore è il numero di nodi interni P o Q; a parità di nodi P, diminuisce al crescere della lunghezza dei nodi Q.

Proposizione 4.1. *Siano S e T due sequenze di nucleotidi che contengono coppie di sottostringhe comuni con caratteri ripetuti. Espandendo i multinsiemi, il* minimal consensus PQ Tree *può essere costruito solo dalle permutazioni in cui è maggiormente preservata la contiguità dei caratteri ripetuti nelle sottostringhe comuni.*

Questa osservazione è importante soprattutto per l'interpretazione dei risultati ottenuti nelle analisi. Infatti *minimal consensus PQ Tree* è stato introdotto proprio per costruire una notazione massimale per famiglie di permutazioni Π. Indipendentemente dal numero di permutazioni $|\Pi|$, la notazione massimale illustra sinteticamente, in spazio $\Theta(n)$ se n è la lunghezza di ciascuna permutazione, la struttura permutativa interna di Π. Se la notazione massimale è quella dell'albero minimale, avremo anche ristretto il numero di pattern che occorre interpretare nella rappresentazione grafica dell'albero (v. 2.5.1.1).

Per ottenere l'albero minimale, nei casi di numerose ripetizioni, è possibile restringere l'applicazione della funzione $REPLACE()$ solo a quegli insiemi di permutazioni che preservano le sottostringhe comuni anche nella codifica numerica. In altre parole, le codifiche in cui si preservano le sottostringhe comuni, non fanno altro che "suggerire" i nodi Q più idonei alla costruzione dell'albero minimale.

Possiamo ora tentare di *costruire* un parziale sottoinsieme delle permutazioni "candidate" a generare alberi minimali codificando le sequenze in accordo alla proposizione 4.1.

4.1 Obiettivi

Siano $\mathcal{A}$ e $\mathcal{B}$ due genomi di nucleotidi nei quali vogliamo cercare le sottostringhe comuni: tali sottostringhe saranno trattate in modo da preservarne la contiguità dei caratteri anche nella codifica numerica, indicandoli quindi come nodi Q nella costruzione dell'albero minimale. Il problema ricorda quello di copertura, ma in questo caso quasi tutti gli intervalli sono sovrapponibili, mentre l'obiettivo resta di riuscire a ricoprire la maggior parte della sequenza con intervalli contigui. Non è detto che i segmenti comuni possano ricoprire per intero le rispettive sequenze, specialmente quando vogliamo che i segmenti comuni abbiano lunghezze maggiori di 1. Le regioni non ricoperte da segmenti comuni potranno essere numerate in modo meno strutturato.

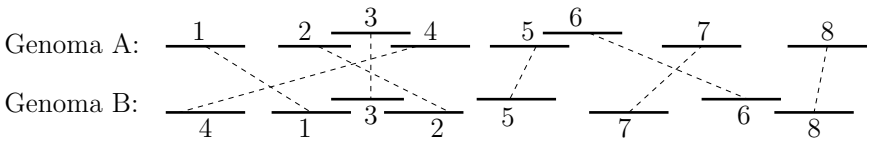


Figura 4.1: Una possibile configurazione di sottostringhe comuni ai due genomi. La forte ridondanza determina sovrapposizioni multiple in cui sono condivisi gli *stessi* segmenti biologici.

Definizione 4.1. *Sia $\mathcal{G}$ un genoma composto da stringhe sull'alfabeto Σ. Una sottostringa S è un* segmento comune *se ogni sua apparizione conserva la stessa successione degli stessi caratteri con segno.*

La richiesta che si preservi il segno può essere rilassata a discrezione del ricercatore qualora interessino palindrome biologiche o palindrome su stringhe.

Si inizia la ricerca delle coppie di sottostringhe comuni di lunghezza maggiore, per poi passare alle lunghezze inferiori. Senza perdita di generalità, associamo la permutazione identità π_{id} al primo genoma $\mathcal{A}$ e, per ogni coppia comune individuata, si enumerano corrispondentemente i segmenti in entrambi i genomi $\mathcal{A}$ e $\mathcal{B}$.

Data l'alta ridondanza, molti segmenti comuni si possono presentare sovrapposti. In questi casi occorre tenere in considerazione che i segmenti sovrapposti, qualora richiedano numerazione differenti, di fatto fanno sempre riferimento alle stesse sottosequenze biologiche (§ 3.1.4). Per i segmenti individuati, codificati numericamente come permutazioni, si ha (Proposizione 2.1) che l'unione, l'intersezione e la differenza dei due segmenti sovrapposti sono ancora segmenti comuni. In Figura 4.1 è mostrato uno scenario tipico in cui i segmenti, solo per semplicità, hanno tutti la stessa lunghezza. Vediamo come si procede in questi casi.

4.1.1 Coppie di segmenti su permutazioni di genomi

Siano $\mathcal{A}$ e $\mathcal{B}$ permutazioni degli stessi caratteri, sui quali cerchiamo le coppie di segmenti comuni. Molto frequenti sono i casi in cui un intervallo viene identifi-

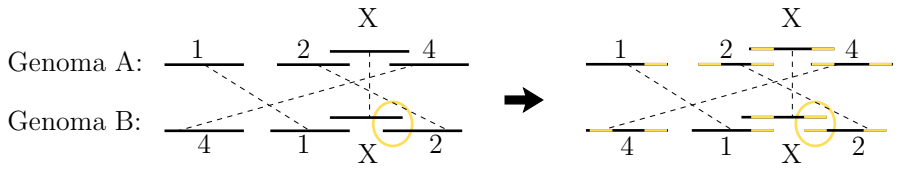


Figura 4.2: L'introduzione della nuova coppia di segmenti X, porta ad evidenziare, partendo dal bordo cerchiato, alcune fra tutte le ripetizioni interne dei segmenti.

cato all'interno di uno più grande. Questa evenienza non è un problema per la relativa copia sull'altro genoma in quanto anch'essa risulterà inclusa in un analogo intervallo: la nuova coppia praticamente non esiste in quanto inglobata in un altra.

Più frequente è il caso mostrato in Figura 4.2 in cui si verifica una sovrapposizione fra più di un intervallo già numerato. Supponiamo di stare identificando coppie di segmenti comuni di lunghezza 3: questo significa che abbiamo già trovato quelli di lunghezza 4 e che dopo questa fase, descritta con un esempio, si passerà a cercare quelli di lunghezza 2. Rapresentiamo testualmente la Figura 4.2. Siano $\mathcal{A}$ e $\mathcal{B}$ i due filamente da analizzare. Le `x` rappresentano intervalli composti dai 4 caratteri {A, C, G, T} della sequenza di DNA, mentre le cifre (esadecimali nel caso ne servano più di 10) indicano quegli intervalli a cui è già stato assegnato un ordinamento numerico per coppie di segmenti individuate in precedenza:

Genoma $\mathcal{A}$ ... `xxx123xx567xx` ...

Genoma $\mathcal{B}$ ... `xxx890xxx` ... `xx567xx` ...

Sul Genoma $\mathcal{A}$ è già stata identificata la stringa `123`, il cui corrispettivo nel Genoma $\mathcal{B}$ si trova in una porzione non visibile nell'esempio; analogo discorso per il corrispettivo `890` del Genoma $\mathcal{B}$. Il segmento comune `567` è visibile in entrambi i filamenti. I conflitti fra le differenti numerazioni vengono risolti con opportune operazioni fra intervalli assumendo una funzione obiettivo F che massimizzi la lunghezza dei segmenti comuni; questo è in linea con la Proposizione 4.1. Le par-

ti non numerate nel genoma $\mathcal{B}$ prendono la numerazione dal rispettivo segmento di riferimento (permutazione identità π_{id}) non ancora assegnato in $\mathcal{A}$, mentre per i conflitti si fa attenzione a non rinumerare i segmenti precedenti per non accorciarne la dimensione originale o per non incorrere in rinumerazioni di altri segmenti. Ora viene identificato un opportuno nuovo segmento biologico comune ai due filamenti, sia `TYZ`.

Genoma $\mathcal{A}$ `... xxx123xx567xx ...`
`TYZ`

Genoma $\mathcal{B}$ `... xxx890xxx ... xx567xx ...`
`TYZ`

Questo indica come le posizioni sovrapposte siano segmenti biologicamente uguali. La funzione obiettivo, mantenendo integri i segmenti già trovati, rispetta l'ordinamento biologico senza introdurre frammentarietà di fatto non presente sul genoma. Massimizzando le lunghezza delle sequenze evitiamo, per esempio, di rinumerare il segmento `89` dalla corrispondenza `YZ-x5`, che produrrebbe frammentazioni nella numerazione (ma non nella sequenza nucleotidica). Invece il segmento `T` si inserisce in entrambi i genomi su intervalli ancora da numerare:

Genoma $\mathcal{A}$ `... xxxx123Tx567xx ...`
`YZ`

Genoma $\mathcal{B}$ `... xxxT890xxx ... xx567xx ...`
`YZ`

Supponiamo di poter assegnare al gruppo `T` il numero `4`. Abbiamo visto per il segmento `YZ` che non è conveniente cambiare la numerazione. Proviamo invece ad inserire un nuovo simbolo per il conflitto fra le numerazioni `Z-9-5`: sia `A` tale nuova cifra; inserita al posto delle 3 occorrenze, introdurrebbe un duplicato nel genoma $\mathcal{B}$; inserita solo in due delle tre, danneggerebbe le corrispondenze fra gli intervalli esistenti o non sarebbe utile (`567` può diventare `A67`).

Genoma $\mathcal{A}$ ... xxxx1234x<u>A</u>67xx ...

Genoma $\mathcal{B}$ ... xxx48<u>A</u>0xxx ... xx<u>A</u>67xx ...

L'unica scelta conveniente rispetto alla funzione obiettivo è quella di ignorare la sovrapposizione di Z ottenendo:

Genoma $\mathcal{A}$ ... xxxx1234x567xx ...

Genoma $\mathcal{B}$ ... xxx4890xxx ... xx567xx ...

Il gap fra i segmenti 1234 e 56, data l'alta ridondanza potrà essere colmato dai segmenti comuni più piccoli: se i due genomi sono permutazioni reciproche tale corrispondenza per quella x verrà trovata sicuramente. Il caso in cui non siano permutazioni reciproche viene discusso nella prossima sezione.

4.1.2 Caso generale

Finora abbiamo ipotizzato di cercare coppie uniche su permutazioni. Dagli esperimenti si deduce tuttavia che le occorrenze di un segmento comune possono essere spesso più di due e non sempre a coppie. Inoltre le sequenze che si analizzano, a meno di casi prodotti ad arte, non presentano né la stessa lunghezza né risultano essere necessariamente permutazioni reciproche, e quindi non presentano lo stesso numero occorrenze di ciascun carattere.

4.1.2.1 Occorrenze multiple di coppie

Occorrenze di un segmento che appare lo stesso numero di volte in entrambi i genomi si gestisce come visto in precedenza con le operazioni di enumerazioni guidata dalla funzione obiettivo; l'unica osservazione in questo caso è che, nonostante siano tutte coppie identiche, occorre trattarle come tutte distinte in modo da generare una permutazione di numeri. Inoltre è possibile scegliere come accoppiarle e questo è uno dei parametri che può essere inserito nella funzione obiettivo in base alle indicazioni del biologo. Si osservi che spesso tali coppie appaiono so-

vrapposte: tipicamente sono sequenze molto ridondanti, tanto da presentare un pattern che si sovrappone a se stesso.

Un problema interessante è quello di gestire le occorrenze multiple preservandone però l'identità che, in questo caso, viene persa. La notazione massimale per i π–pattern può rappresentare adeguatamente anche permutazioni con duplicati, dunque un'indagine in quel senso è da rivolgere su una possibile variante degli alberi PQ, e del relativo algoritmo di costruzione, basato sugli intervalli irriducibili di permutazioni senza ripetizioni. D'altro canto non è immediatamente riconducibile ad approcci di *multiple sequence alignment* o di *match chaining* che classicamente non prevedono una rigorosa strutturazione interna come avviene invece nel caso di permutazioni numeriche.

4.1.2.2 Occorrenze non accoppiate

Altre volte si trovano molte più occorrenze di un segmento su un genoma piuttosto che sull'altro. Si possono fare considerazioni analoghe al caso di occorrenze multiple e, dopo aver implementato una politica per associare le possibili coppie, i segmenti rimanenti semplicemente vengono tralasciati per essere quasi certamente elaborati durante l'individuazione dei segmenti comuni più brevi.

4.1.2.3 Sequenze biologiche

Le sequenze biologiche non è detto che abbiano la stessa lunghezza o che presentino la stessa quantità di occorrenze di ciascun carattere. In questi casi, dopo avere ricercato e numerato i segmenti comuni da quelli più lunghi a quelli più brevi, anche dopo avere analizzato le occorrenze dei singoletti, ci si ritrova con buchi di sequenza non numerata. L'idea per riuscire a trattare formalmente queste sequenze è di trasformarle in permutazioni. Si potrebbe procedere eliminando opportuni caratteri dalla sequenza più lunga o, piuttosto, apportando interventi che non alterino le caratteristiche permutative originali.

Nel caso di sequenze di lunghezze diverse, si associa la permutazione identità π_{id} a quella più lunga: in questo modo abbiamo maggiori garanzie di non restare

"a corto" di cifre per la numerazione della seconda stringa. Tuttavia non ne abbiamo la certezza e, dopo aver enumerato tutti i possibili segmenti comuni come descritto precedentemente, in generale si hanno dei buchi in entrambe le sequenze. Si considerino i seguenti due genomi:

Permutazione π_{id}	1	2	3	4	5	6	7	8
Genoma $\mathcal{A}$	A	A	A	A	C	C	G	G
Genoma $\mathcal{B}$	A	A	C	C	C	T	G	

Usiamo la permutazione identità pari alla lunghezza massima delle due sequenze come rifermento per la numerazione. Supponiamo che la funzione obiettivo sia tale per cui vengono selezionati i segmenti comuni che seguono:

Permutazione π_{id}	1	2	3	4	5	6	7	8
Genoma $\mathcal{A}$	A	A	A	A	C	C	G	G
Genoma $\mathcal{B}$	A	A	C	C	C	T	G	

Rispetto alla numerazione di riferimento π_{id} otteniamo:

Permutazione π_{id}	1	2	3	4	5	6	7	8
Genoma $\mathcal{A}$	A	A	A	A	C	C	G	G
	_	2	3	_	5	6	_	8
Genoma $\mathcal{B}$	A	A	C	C	C	T	G	
	2	3	5	6	_	_	8	

Vi sono delle posizioni in cui non siamo riusciti ad inserire alcuna corrispondenza. Per rendere le due stringhe una permutazione dell'altra procediamo marcando con un punto le cifre per le quali non esiste una corrispondenza nell'altra sequenza. Per il genoma $\mathcal{B}$ si usano numeri univoci mai usati nella permutazione di riferimento π_{id}:

Permutazione π_{id}	1	2	3	4	5	6	7	8
	1̣	2	3	**4̣**	5	6	**7̣**	8
	2	3	5	6	**9̣**	**0̣**	8	

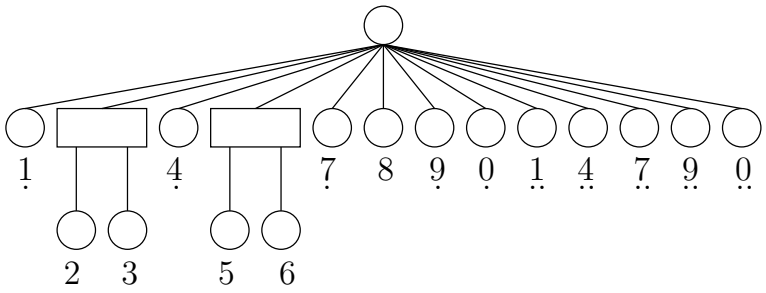


Figura 4.3: I sottoalberi senza cifre marcate sono gli unici π–pattern massimali: $(2-3)$ e $(5-6)$. Si noti che la cifra 8 è un sigoletto.

Le nuove cifre $\dot{9}$ e $\dot{0}$ sono state usate nel genoma $\mathcal{B}$. Ora, in entrambe le sequenze, ovunque appaia una cifra marcata $\dot{c}$, la sostituiamo con $\dot{c}\ddot{c}$, mentre nella sequenza in cui non appare $\dot{c}$, inseriamo $\dot{c}$ all'inizio e $\ddot{c}$ alla fine. Applichiamo prima queste operazioni alle cifre $\dot{c}$ marcate in $\mathcal{A}$:

Genoma $\mathcal{A}$	**1̇**	**1̤**	2	3	**4̇**	**4̤**	5	6	**7̇**	**7̤**	8	
Genoma $\mathcal{B}$	**1̇**	**4̇**	**7̇**	2	3	5	6	**9̇**	**0̇**	8	**7̤**	**4̤**	**1̤**

Applicando la stessa procedura anche per le cifre $\dot{c}$ in $\mathcal{B}$, otteniamo le due permutazioni:

Genoma $\mathcal{A}$	9̇	0̇	1̇	1̤	2	3	4̇	4̤	5	6	7̇	7̤	8	0̤	9̤
Genoma $\mathcal{B}$	1̇	4̇	7̇	2	3	5	6	9̇	9̤	0̇	0̤	8	7̤	4̤	1̤

Le due sequenze marcate sono permutazioni reciproche, nelle quali sono presenti tutti i π–pattern che apparivano nelle sequenze originali. Invece tutti i π–pattern che appaiono nelle sequenze marcate ma non appaiono in quelle originali devono necessariamente contenere una delle cifre marcate (vedi Figura 4.3). Infatti l'inserimento delle cifre marcate $\dot{c}$ e $\ddot{c}$, non trovando corrispondenza con la sequenza reale, viene operato usando la Proposizione 4.1 in senso inverso: la separazione delle cifre $\dot{c}$ da quelle $\ddot{c}$ non può creare nodi Q ma solo nodi P, evitando la costruzione di strutture apparentemente interessanti ma inesistenti.

Genoma $\mathcal{A}$			A		**A**	**A**	A		**C**	**C**	G		**G**		
	$\dot{9}$	$\dot{0}$	$\dot{1}$	$\ddot{1}$	2	3	$\dot{4}$	$\ddot{4}$	5	6	$\dot{7}$	$\ddot{7}$	8	$\ddot{0}$	$\ddot{9}$
Genoma $\mathcal{B}$				**A**	**A**	**C**	**C**	C		T		**G**			
	$\dot{1}$	$\dot{4}$	$\dot{7}$	2	3	5	6	$\dot{9}$	$\ddot{9}$	$\dot{0}$	$\ddot{0}$	8	$\ddot{7}$	$\ddot{4}$	$\ddot{1}$

Costruito l'albero PQ, i sottoalberi che non contengono cifre marcate sono i soli π–pattern massimali delle sequenze originali: $(A-A)$, $(C-C)$ e il singoletto (G). Infatti, se un certo sottoalbero T_i di un *minimal consensus PQ Tree* T non ha cifre marcate e, se T_i *non* è sottoalbero di altri sottoalberi con cifre marcate, allora T_i rappresenta un π–pattern massimale.

Questa costruzione, pur basandosi sullo studio formale di permutazioni esatte, effettivamente permette di modellare sequenze con un certo numero di errori o variabilità nel genoma. Tuttavia gli elementi di base su cui opera sono dei segmenti che devono essere uguali. Un utile miglioramento su cui indagare, può essere quello di processare, prima di applicare questa codifica, le due sequenze in modo da definire dei segmenti che siano "uguali a meno di errori", verosimilmente usando un concetto di *edit distance*, quale la *Levenshtein distance*, anche sui segmenti comuni.

4.2 Suffix tree

Un *suffix tree* [32, 33, 34] è una struttura dati ad albero che evidenzia la struttura interna di una stringa esponendone tutti i suffissi.

Un *suffix tree* $\mathcal{T}$ per una stringa S di lunghezza n è un albero radicato con esattamente n foglie numerate da 1 ad n. Ogni nodo interno, esclusa al più la radice, ha almeno due figli e ogni arco è etichettato con una sottostringa di S. Due archi uscenti dallo stesso nodo non possono avere etichette che iniziano con lo stesso carattere. La caratteristica fondamentale è che la concatenazione delle etichette lungo il cammino dalla radice alla foglia i è esattamente il suffisso $S[i, n]$ di S di lunghezza $n - i + 1$.

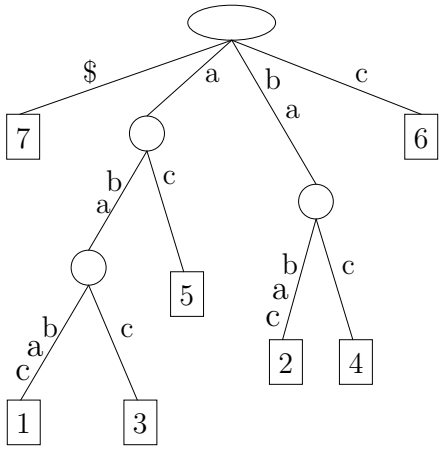


Figura 4.4: Il *suffix tree* per la stringa “ababac$”.

Dal momento che tutti i nodi interni hanno almeno 2 archi uscenti, nel caso peggiore ci possono essere $n - 1$ di tali nodi, oltre alla radice, per un totale di $n + (n - 1) + 1 = 2n$ nodi.

In Figura 4.4 è illustrato l’albero dei suffissi per la stringa “ababac”. Lungo il cammino dalla radice alla foglia 1 si legge l’intera stringa “ababac” mentre lungo il cammino dalla radice alla foglia 2 si legge il suffisso “babac” che inizia nella posizione 2 della stringa.

La definizione precedente non garantisce l’esistenza di un albero dei suffissi per ogni stringa. Il problema è che se qualche suffisso è prefisso di un altro suffisso di S il cammino relativo a tale suffisso non termina in una foglia. Ad esempio se togliamo l’ultimo carattere c alla stringa “ababac” ottenendo la stringa “ababa”, il suffisso *ba* è prefisso di “baba” e quindi il cammino relativo a *ba* non termina in una foglia, ma termina nel punto intermedio “ba␣ba” lungo il percorso verso le foglie 2 e 4.

Per evitare questo problema si assume che ogni stringa S termini con un carattere diverso da tutti gli altri caratteri della stringa in modo che nessun suffisso possa essere prefisso di un altro suffisso. Se ciò non è vero possiamo sempre aggiungere alla fine della stringa un carattere terminatore che non appartiene

all'alfabeto su cui è costruita la stringa. Tipicamente ci si riferisce a questo carattere con \$, come si può vedere in Figura 4.4.

L'etichetta di un cammino dalla radice ad un nodo (interno o foglia) è la concatenazione delle etichette degli archi del cammino. L'etichetta di un nodo è l'etichetta del cammino dalla radice a quel nodo. In particolare l'etichetta della radice è la stringa nulla e l'etichetta di una foglia è il suffisso associato a tale foglia.

4.2.1 Applicazioni

I *suffix tree* permettono di risolvere il problema del matching esatto in tempo lineare ma la loro principale virtù è che possono essere usati per risolvere in tempo lineare molti altri problemi più complessi. L'applicazione classica dei *suffix tree* è il problema della sottostringa: dato un testo T di lunghezza n, dopo la costruzione del *suffix tree*, che richiede tempo $\Theta(n)$, è possibile rispondere in tempo $\mathcal{O}(m)$ alla domanda se una stringa S di lunghezza m compare in T semplicemente cercando di fare il matching della stringa S lungo uno dei rami dell'albero. La stringa S è contenuta nel testo T se S si esaurisce ad un certo punto lungo uno dei percorsi dell'albero, ed occorre a tutte le posizioni individuate dalle foglie sottostanti quel punto.

Fra i problemi che si possono risolvere in tempo lineare usando i *suffix tree* ricordiamo [35]:

- trovare tutte le occorrenze di una famiglia di stringhe;
- trovare tutte le sottostringhe comuni a una famiglia di stringhe;
- trovare tutte le palindromi massimali;
- trovare tutti i tandem repeats;
- ricerca con errori, particolarmente efficiente per piccoli alfabeti.

Noi useremo i *suffix tree* per trovare le sottostringhe comuni.

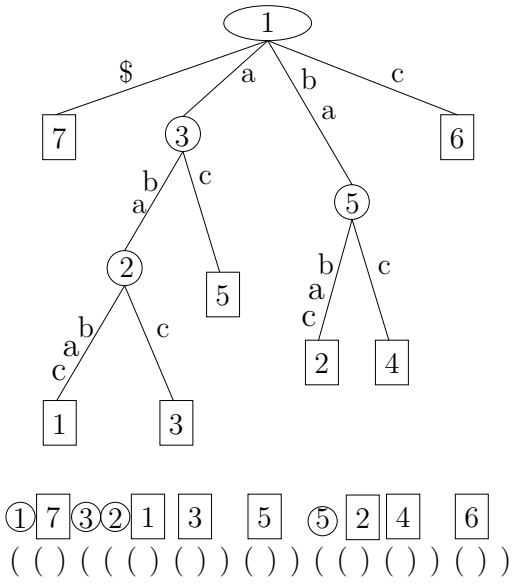


Figura 4.5: Il *suffix tree* per la stringa "ababac$" con la rappresentazione associata a parentesi bilanciate.

4.2.2 Suffix tree compressi

Nonostante la grossa efficienza computazionale, la rappresentazione dei *suffix tree* in memoria è un parametro critico, dal momento che occorrono da 10 a 30 bytes per ciascun elemento della stringa [33, 36, 37, 38]. Questo significa che per una sequenza di DNA, in cui ogni elemento usa di fatto 2 bit, si ottengono *suffix tree* con una dimensione da 40 a 120 volte quella dell'input: il genoma umano, contenuto in 800MB, richiede un *suffix tree* che occupa almeno 30GB di memoria. Per questo motivo, ormai da alcuni anni, sono state proposte varie soluzioni [39, 40], che riducono drasticamente tali requisiti, introducendo i *suffix tree* compressi.

Recentemente Sadakane [41], dopo aver identificato tutte le operazione astratte che deve supportare la struttura dati dei *suffix tree*, ha fornito una soluzione in cui ogni operazione è costante o, solo in pochi casi, $\mathcal{O}(\log n)$. Lo spazio occupato è $|CSA|+6n+o(n)$ bit dove $|CSA|$ è lo spazio occupato dal *suffix array* usato. La soluzione di Sadakane usa le parentesi bilanciate [42] per rappresentare il *suffix*

tree (v. Fig. 4.5). La rappresentazione è prodotta da una visita anticipata, in cui si stampa '(' quando si incontra un nodo per la prima volta, e si stampa ')' visitando il nodo per l'ultima volta.

La proposta di Sadakana è stata implementata in [43] ottenendo un'occupazione in spazio pari a $\mathcal{O}(n \log |\Sigma|)$ bit dove Σ è l'alfabeto. Tale implementazione è stata usata in questa tesi per condurre alcuni dei test.

4.3 L'algoritmo

Per poter ricercare i segmenti comuni nelle due sequenze, prima si costruisce il *suffix tree* $\mathcal{T}$ relativo alla concatenazione dei due genomi $\mathcal{A}\$\mathcal{B}\#$ di lunghezza complessiva n in tempo $\mathcal{O}(n)$. Ogni foglia dell'albero rappresenta dunque un suffisso che si trova in una delle due sequenze oppure in entrambe le sequenze. L'etichetta di un nodo che contiene foglie in entrambe le sequenze è dunque una stringa comune ad $\mathcal{A}$ e $\mathcal{B}$. Se una stringa α è comune ad entrambe le sequenze, allora α etichetta uno dei nodi interni u di $\mathcal{T}$, con occorrenze individuate alle posizioni delle foglie che discendono da u. Si processa $\mathcal{T}$ marcando i nodi come appartenenti all'una o all'altra sequenza in relazione alla posizione delle foglie nel loro sottoalbero.

Algoritmo 4.1 Segmenti comuni

```
1:  i ← n                                   // n è l'ultima parentesi bilanciata
2:  while i > 0 do
3:    if i è una foglia then
4:      if i < |A| then
5:        Marca la foglia i ∈ A
6:      else
7:        Marca la foglia i ∈ B
8:      end if
9:    end if
10:   Marca il nodo padre(i) come i
11:   i ← i − 1
12: end while
```

Usando la rappresentazione a parentesi bilanciate (v. Figura 4.5) è sufficiente scorrere la lista di parentesi dall'ultima verso la prima: questo equivale ad una

visita anticipata a ritroso, e garantisce che quando si raggiunge un nodo, tutti i sui discendenti sono già stati elaborati.

Alla riga `10`, si marca il padre dell'indice corrente i come appartente alla stessa sequenza cui appartiene i: questo permette successivamente di distinguere i nodi che hanno discendenti in una o in entrambe le sequenze. Si noti che si marca il padre sia delle foglie sia dei nodi, ricavando così un albero in cui ogni nodo dà informazioni sui segmenti comuni dei propri discendenti.

Dal momento che tutti i segmenti comuni sono da ricercarsi sulle etichette dei nodi e, allo stesso tempo, vogliamo trovarli dal più lungo al più corto, allora occorre effettuare una visita *bottom-up* dei nodi di $\mathcal{T}$ che abbiano discendenti in entrambe le sequenze. È possibile ideare un algoritmo per la visita bottom-up di un grafo che abbia complessità lineare.

Una procedura intuitiva, ma ancora efficiente, consiste nel fare una visita in tempo $\mathcal{O}(n)$ dell'albero, scorrendo le parentesi bilanciate e inserendo i nodi candidati in una opportuna struttura dati. Al termine della visita si ordina in tempo $\mathcal{O}(n)$ rispetto alla profondità dei nodi usando un algoritmo di *Radix-sorting*: infatti sarà nota la lunghezza massima delle stringhe comuni.

Algoritmo 4.2 Preparazione alla visita "bottom-up"

```
1: for all i in ParentesiBilanciate do
2:    if i è un nodo and i ∈ A ∧ B  then        // i contiene segmenti comuni
3:       Insert(i, Nodes)
4:    end if
5: end for
6: Radix-sort(Nodes)
```

Scorrere gli elementi, così ordinati per lunghezza decrescente, equivale ad una visita *bottom-up* in cui si analizzano tutti i nodi a profondità h prima di passare a quelli di profondità $h-1$. Sia v il nodo estratto da $Nodes$ con etichetta α. Tutte le foglie del sottoalbero di v individuano un'occorrenza del segmento comune α. Ogni occorrenza è caratterizzata dall'indice di inizio, corrispondente al valore della foglia f, e dalla lunghezza del segmento, corrispondente alla profondità $h = |\alpha|$ del nodo v. In generale ci siano j occorrenze in $\mathcal{A}$ e k occorrenze in $\mathcal{B}$, con $j, k \geq 1$. Questo significa che ci sono $j + k$ foglie nel sottoalbero di v.

L'euristica può allocare quei segmenti che vanno a compattarsi con altri già presenti o quelli che coprono più sovrapposizioni fra segmenti identici oppure applicare opportune politiche biologiche di allocazione fra le coppie, come visto al § 4.1.1. Al termine di questa fase, se $j = k$ avremo allocato tutte le coppie di segmenti. Se invece $j \neq k$, significa che ci sono più occorrenze di α in una delle due sequenze. Usando il modello permutativo non possiamo introdurre duplicati e dunque di fatto non possiamo assegnare una numerazione ai segmenti che non hanno un corrispettivo nell'altro genoma. Per questo motivo prepariamo due liste $\ell_{\mathcal{A}}$ e $\ell_{\mathcal{B}}$ in cui inseriamo le occorrenze, rispettivamente di $\mathcal{A}$ e $\mathcal{B}$, che non sono state numerate al nodo v, passandole al padre v'. Portando queste foglie al livello superiore, candidiamo le rispettive occorrenze a essere numerate da segmenti di lunghezza inferiore, statisticamente più numerosi e con più probabilità di essere associati. Questa operazione, facendo uso di semplici concatenamenti fra liste in tempo $\mathcal{O}(1)$, non innalza la complessità lineare dell'algoritmo e fa sì che le foglie ancora da numerare nel sottoalbero possano essere mantenute fra i discendenti immediati.

Algoritmo 4.3 Codifica

```
1: for all i in Nodes do
2:     Usa l'euristica per numerare i segmenti alle foglie di i.
3:     if i ha ancora foglie then
4:         Inserisci le foglie f_A in ℓ_A
5:         Inserisci le foglie f_B in ℓ_B
6:         Appendi ℓ_A e ℓ_B al padre(i)
7:     end if
8: end for
```

Questo è ancora vero non solo per v ma per tutti i nodi antenati che vanno dalla foglia f fino alla radice. In altre parole, ciascuna foglia può essere l'inizio di tanti segmenti comuni sovrapposti, ciascuno di lunghezza diversa, quanti sono i nodi che vanno dalla radice alla foglia f. Per questo motivo le foglie non selezionate per la numerazione ad uno nodo dovranno essere scandite al nodo superiore. Questo innalzerebbe la complessità a meno di non tenerle discendenti immediate. L'uso di un algoritmo *greedy* che cerchi subito di allocare il segmento

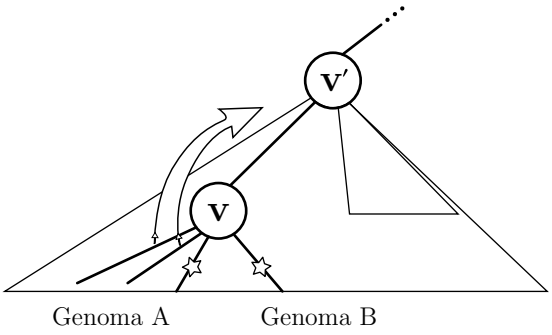


Figura 4.6: Dopo aver numerato le possibili coppie si portano al nodo superiore le occorrenze singole.

più lungo disponibile per quella foglia, massimizzando la funzione obiettivo, ci permette di rimuovere la foglia da ulteriori elaborazioni.

4.4 L'euristica

L'euristica può allocare quei segmenti che vanno a compattarsi con altri già presenti o quelli che coprono più sovrapposizioni fra segmenti identici oppure applicare opportune politiche biologiche di allocazione fra le coppie. Si noti che nonostante l'allocazione proceda dai segmenti più lunghi a quelli più brevi, l'alto numero di sovrapposizioni porta ad avere molte intersezioni che è possibile gestire in diversi modi.

Si fa solo un esempio. Al § 4.1.2.3 si è fatto un esempio in cui l'euristica ha associato i segmenti nel seguente modo:

Permutazione π_{id}	1	2	3	4	5	6	7	8
Genoma $\mathcal{A}$	A	A	A	A	C	C	G	G
Genoma $\mathcal{B}$	A	A	C	C	C	T	G	

Se l'euristica della funzione obiettivo viene costruita piuttosto in modo da scegliere i segmenti alle estremità, cercando di massimizzare dunque la contiguità con altri segmenti, si avrebbe avuto:

Permutazione π_{id}	1	2	3	4	5	6	7	8
Genoma $\mathcal{A}$	A	A	A	A	C	C	G	G
Genoma $\mathcal{B}$	A	A	C	C	C	T	G	

In questo caso si sarebbe trovato l'unico albero minimale, ottenendo la notazione massimale $(3-4-5-6)$ invece di $(2-3)$ e $(5-6)$.

4.5 Complessità

Come già evidenziato nel testo, la ricerca di tutti i segmenti comuni può essere fatta in tempo $\mathcal{O}(n)$ con $n = |\mathcal{A} + \mathcal{B}|$. Ad ogni passo viene considerata una sottostringa α e le relative occorrenze individuate da un certo numero di foglie. La numerazione delle occorrenze avviene numerando, una sola volta, un insieme di caratteri delle sequenze originali e porta alla rimozione delle foglie f che ne indicano la posizione d'inizio.

Potrebbe apparire che l'euristica possa innalzare la complessità in relazione al numero di segmenti comuni duplicati da elaborare. Si noti infatti che in un genoma di n caratteri identici ci sono $\mathcal{O}(n^4)$ segmenti comuni.

Questo problema viene evitato elaborando i segmenti in ordine decrescente di lunghezza: le occorrenze più lunghe inglobano le più brevi. Naturalmente al diminuire della lunghezza occorre fare i controlli dei segmenti che si sovrappongono, infatti nell'esempio:

```
Genoma A    ... xxx123xx567xx ...
                    TYZ

Genoma B    ... xxx890xxx ... xx567xx ...
                  TYZ
```

per arrivare a:

```
Genoma A    ...  xxxx1234x567xx ...
Genoma B    ...  xxx4890xxx ...  xx567xx ...
```

occorrerebbe scandire i segmenti già numerati per determinare in quali punti del segmento comune è possibile inserire la numerazione senza danneggiare le precedenti più lunghe.

A questo prosito basta aggiungere, in tempo costante, affianco alla numerazione della codifica anche l'indice al quale finisce la numerazione di questo segmento e rendere disponibili, agli estremi destro e sinistro di ogni segmento, l'indice dell'estremità opposta. In questo modo, se l'euristica non altera la numerazione dei segmenti già codificati, si riesce ad esaminare sempre e solo quelle posizioni che ancora non hanno ricevuto un numero di codifica.

Conclusioni

Le ripetizioni genomiche restano ancora uno dei grossi problemi da risolvere per gli approcci formali.

Formalizzare le ripetizioni con il modello dei multinsiemi nell'ambito dell'analisi genomica può essere utile solo nel caso pochissime duplicazioni, specialmente in analisi genomiche ad alto livello. Soprattutto a livello nucleico, i risultati sembrano confermare che non è tanto importante l'*ordine* con cui quei caratteri si ripetono, quanto piuttosto il *contesto* in cui si ripetono. Numerando con degli ordinali ogni ripetizione, le possibili disposizioni di tali caratteri ripetuti lasciano inalterata la sequenza genomica biologica, ma producono permutazioni numeriche in quantità fattoriale.

Una strada alternativa, che non impone una struttura interna ai pattern, consiste nel preservare tutte le ripetizioni utilizzando il concetto di *fingerprint*. Ogni *fingerprint* individua sulle sequenze biologiche uno o più intervalli massimali, ovvero intervalli che sono delimitati a destra e a sinistra da un carattere non appartente al *fingerprint*, oppure l'estremità della sequenza. Tuttavia la mancanza di una struttura interna rende meno agevole l'applicazione di metodi formali.

La metodologia proposta ricerca i segmenti comuni, procedendo nella costruzione di permutazioni numeriche candidate a generare i *minimal consensus PQ Tree*, ovvero alberi PQ con il minor numero di permutazioni della loro frontiera. La proposta si muove nella giusta direzione, pur non fornendo una soluzione definitiva ad un problema formale, che può ancora essere analizzato sotto altri punti di vista.

- Abbiamo visto l'algoritmo usato per creare gli intervalli di tipo Q, in una visita del suffix tree: è possibile mettere in relazione particolari visite dei Suffix Tree con la struttura degli alberi PQ?

- La notazione massimale può rappresentare π–pattern con elementi duplicati, mentre occorrerebbe investigare se è possibile costruire una variante dell'albero PQ in cui i nodi Q possano rappresentare segmenti contigui identici, ma duplicati (p.e. usando una nozione di "colore").

- Si è visto come applicare l'analisi della struttura permutativa interna delle stringhe anche a sequenze iniziali di diverso contenuto e lunghezza. Può essere interessante estendere questo tipo di analisi anche ai segmenti comuni che differiscono meno di un certa soglia, modellandone la differenza con una *edit distance*, quale la *Levenshtein distance*.

Appendici

Appendice A

Un caso d'uso

Si vuol chiarire la prassi e le metodologie operative del campo bioinformatico. A questo fine si riporta un caso d'uso reale, adottato nell'architettura Grid di workflow scientifici realizzata in partnership fra ISTI–CNR[1] (Pisa) e ISC–CRA[2] (Fiorenzuola) nell'ambito del progetto ESCOGITARE [44].

Il lettore più interessato può anche cimentarsi nell'esecuzione dell'esperimento *in silico*: tutti gli strumenti utilizzati sono accessibili online e, insieme al procedimento concettuale, vengono indicati i punti salienti della procedura operativa.

A.1 Detection di Secale cereale via Real time PCR

La necessità di individuare la presenza di segale in alimenti è legata, oltre che alla possibilità di caratterizzare un determinato prodotto, anche all'esigenza di assicurare alimenti adatti per le persone affette da celiachia che devono necessariamente evitare il consumo di frumento, segale, orzo e triticale[3].

[1] Istituto di Scienza e tecnologie dell'Informazione – Consiglio Nazionale delle Ricerche.

[2] Istituto Sperimentale per la Cerealicoltura – Consiglio Nazionale delle ricerche Agroalimentari.

[3] Il triticale è un ibrido artificiale tra la segale e il frumento, creato alla fine del XIX secolo ma solo ultimamente coltivato su larga scala. Esso associa la resistenza al freddo della segale con l'attitudine alla panificazione del frumento. La parola stessa è una fusione delle parole latine Triticum ("frumento") e Secale ("segale")

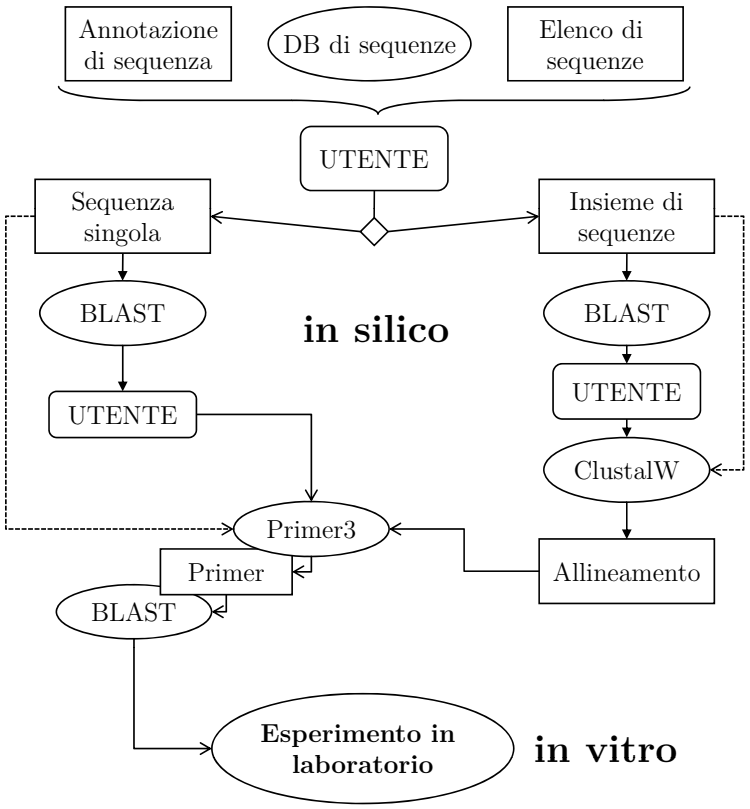


Figura A.1: Lo schema a blocchi che rappresenta il flusso delle operazioni nel caso d'uso.

A.1.1 Individuazione delle sequenze

Occorre prima di tutto individuare i geni che codificano le proteine di glutine, responsabile dell'intolleranza alimentari dei celiaci. Nel caso della segale tale proteina si chiama *secaline*. Facendo ricorso alla GenBank[4], disponibile online[5], si ha la possibilità di cercare le sequenze depositate che siano state annotate come "secaline".

Introducendo il termine – rigorosamente in inglese – *secalin* nello spazio riservato alle ricerche e selezionando "*nucleotide*" si avvia una ricerca sulle sequenze

[4] GenBank è un database di sequenze annotate, pubblicamente accessibile e prodotto al National Center for Biotechnology Information (NCBI) come partecipante al International Nucleotide Sequence Database Collaboration (INSDC).

[5] http://www.ncbi.nlm.nih.gov/.

di nucleotidi disponibili.

Siamo al punto di scelta per l'utente mostrato in Figura A.1. Infatti il risultato della ricerca presenta una lista, più o meno lunga, di sequenze annotate. Occorre la valutazione del ricercatore per determinare quali, e quante, utilizzarne. Spesso ci sono più sequenze simili su cui lavorare ma, in questo caso, una sola delle risposte mostrate contiene l'intera sequenza codificante. Alla voce AF000227 corrisponde la descrizione: "*Secale cereale omega secalin gene, complete cds*". Procediamo con una sola sequenza, seguendo il ramo sinistro in Figura A.1.

A.1.2 Verifica delle sequenze

Individuata una possibile sequenza candidata, occorre verificarne la specificità, ovvero il suo potenziale utilizzo per la successiva detection di segale. Si procede allora ad un'analisi BLAST (§ 2.6.2), ancora online via NCBI.

Dalla precedente lista dei risultati di "secalin", seguendo il link AF000227 si ottiene la sequenza nucleica desiderata. A questo punto si può lanciare un BLAST[6] di verifica sul database *nucleotide collection*.

Sequences producing significant alignments:

Accession	Description	Max score	Coverage	Identities
AF000227.1	Secale cereale omega secalin gene, complete cds	1.705e+04	100%	100%
X60295.1	S.cereale Sec1 gene for omega secalin	3227	19%	98%
X60294.1	S.cereale Sec1 gene for omega secalin	2401	14%	98%
CT009735.1	Triticum aestivum	1474	17%	89%
CT009625.1	Aegilops tauschii	1474	17%	89%
CR626934.1	Triticum aestivum	1474	17%	89%
CR626926.1	Aegilops tauschii	1474	17%	89%
AF459639.1	Triticum monococcum	1456	14%	88%
M36941.1	Hordeum vulgare C-hordein gene, complete cds	1186	21%	84%
AF280606.1	Triticum aestivum omega gliadin pseudogene	1157	22%	77%
AB059812.1	Triticum aestivum pseudogene for omega gliadin	1003	18%	80%

Figura A.2: Il risultato di BLASTn relativo alla sequenza "secalin" direttamente dal sito NCBI. Osservando la colonna *Identities*, si noti anche "quanta parte" di secalin, il glutine della segale, è presente negli altri genomi che, si capisce dai nomi, sono anch'essi cereali.

[6] La variante per ricerche nucleotidiche su {A, C, G, T} si chiama BLASTn.

BLAST restituisce una classifica delle sequenze, fra tutte quelle presenti nel database di NCBI, che meglio corrispondono a quella indicata dal ricercatore. Nella classifica mostrata in Figura A.2, i risultati in cima alla lista sono quelli con lo score più alto e quindi con maggiori corrispondenze. Il primo risultato riporta il 100% di specificità, ed è dunque la nostra scelta.

A.1.3 Design dei primer

Verificata la specificità della sequenza rispetto a tutte quelle nel database, si passa al design dei primer più appropriati mediante il programma Primer3 (v. § 2.6.4).

Ora si seleziona il segmento della sequenza che trova corrispondenza solo su sequenze di segale e si richiede[7] a Primer3 il calcolo dei primer con il comando "pick primers". Il risultato è mostrato in Figura A.3;

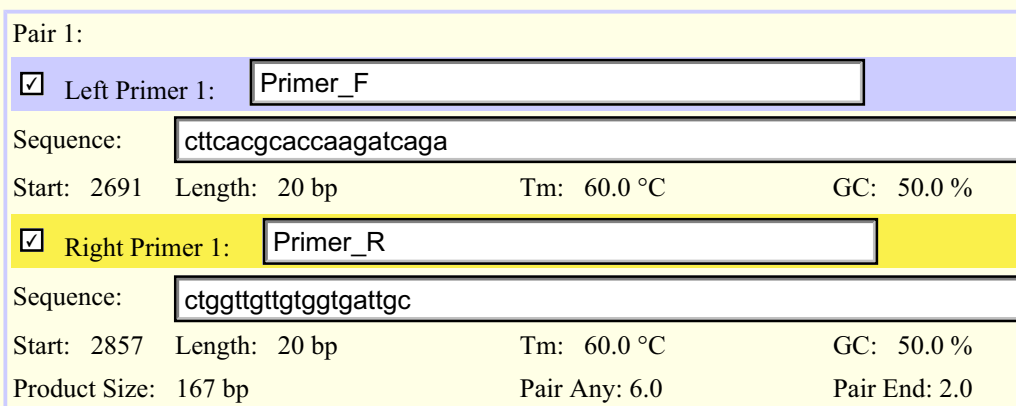


Figura A.3: Sono stati individuati i due segmenti idonei ad innescare la traduzione del DNA: primer destro e sinistro.

A.1.4 Verifica in silico e Test in vitro

Si procede quindi ad un ultimo BLAST di verifica delle due sequenze così ottenute per accertarsi definitivamente che trovino una netta specificità solo le sequenze di segale.

[7] Uno strumento user-friendly si trova all'indirizzo http://www.bioinformatics.nl/cgi-bin/primer3plus/primer3plus.cgi.

Si procede quindi in laboratorio con la reazione a catena della polimerasi (Polymerase Chain Reaction), comunemente nota con l'acronimo PCR. La tecnica consente la moltiplicazione selettiva di frammenti di acidi nucleici dei quali si conoscano le sequenze nucleotidiche iniziali e terminali.

Infatti i primer trovati fin qui *in silico*, vengono ora sintetizzati ed usati come innesco selettivo per la replicazione di sequenze di DNA. Questo consente di ottenere molto rapidamente *in vitro* la quantità di materiale genetico necessaria per successive applicazioni.

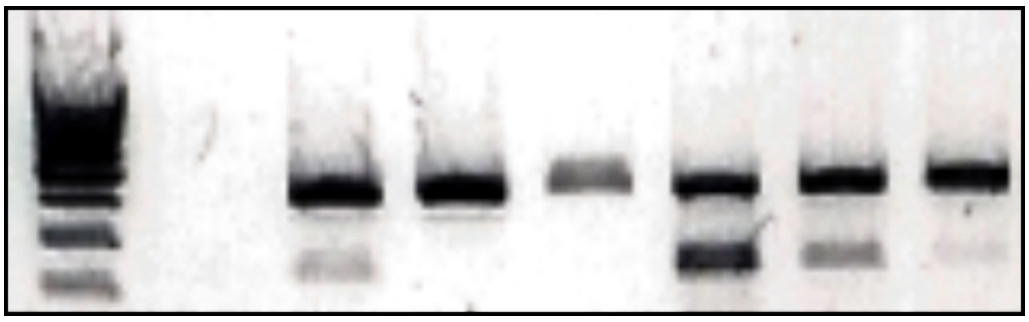


Figura A.4: Dopo circa 25 cicli di sintesi del DNA, il prodotto della PCR include, oltre al DNA iniziale, circa 10^5 copie del DNA selezionato, una quantità sufficiente ad essere visualizzata in bande.

La PCR produce una replicazione esponenziale del DNA e, usando opportuni strumenti di laboratorio, si possono ottenere delle "fotografie" come quella in Figura A.4.

A.1.5 Sequenze multiple

Nella sezione A.1.1, dopo la ricerca della parola *secalin* su GenBank, avevamo scelto un'unica sequenza che rispondeva in pieno al nostra ricerca. In altri casi il ricercatore può decidere di usare più sequenze simili fra loro. Prima di procedere occorre quindi un allineamento multiplo di sequenze, tramite ClustalW (v. § 2.6.3), per localizzare le regioni più conservate. Individuate le sequenze consenso si può andare avanti, lungo il ramo destro della Figura A.1, con gli stessi passaggi descritti per la sequenza singola.

Appendice B

Formato FASTA

FASTA è un formato testuale per rappresentare sequenze di acidi nucleici (o peptidiche), nel quale ogni base azotata (o aminoacido) è rappresentata da una lettera dell'alfabeto.

Un file di testo FASTA inizia con una riga d'intestazione, il cui primo carattere è il segno maggiore '>'. Non devono esserci spazi fra '>' e la prima parola che è l'identificatore della sequenza. Tutto il resto della linea è la descrizione. L'identificare e la descrizione sono entrambi opzionali. Qualunque altra linea che inizia con '>' indica l'inizio di una nuova sequenza e la fine della precedente.

Ogni riga non deve superare gli 80 caratteri. Le sequenze possono essere sequenze di acidi nucleici oppure di proteine e possono contenere *gaps*. Le sequenze vanno rappresentate usando soltanto i codici standard IUB/IUPAC per aminoacidi e acidi nucleici, e sono ammesse le seguenti eccezioni:

1. caratteri minuscoli, comunque interpretati come maiuscoli;
2. il trattino '-' indica i gaps di acidi nucleici;
3. il *dash* '/' indica i gaps di aminoacidi;
4. sono ammessi 'U' e '*' per gli aminoacidi.

Codice Nucleico	Significato
A	**A**denina
C	**C**itosina
G	**G**uanina
T	**T**imidina
U	**U**racile
R	G A (pu**R**ine)
Y	T C (p**Y**rimidine)
K	G T (**K**etone)
M	A C (a**M**ino group)
S	G C (**S**trong interaction)
W	A T (**W**eak interaction)
B	G T C (non A) (**B** viene dopo A)
D	G A T (non C) (**D** viene dopo C)
H	A C T (non G) (**H** viene dopo G)
V	G C A (non T, non U) (**V** viene dopo U)
N	A G C T (a**N**y)
X	masked
-	gap di lunghezza indeterminata

Tabella B.1: Codici IUB/IUPAC per gli acidi nucleici usati nel formato FASTA.

Spesso si è lasciata la forma inglese onde evidenziare la corrispondenza fra simbolo e significato.

Codice Aminoacido	Significato
A	Alanine
B	Aspartic acid or Asparagine
C	Cysteine
D	Aspartic acid
E	Glutamic acid
F	Phenylalanine
G	Glycine
H	Histidine
I	Isoleucine
K	Lysine
L	Leucine
M	Methionine
N	Asparagine
O	Pyrrolysine
P	Proline
Q	Glutamine
R	Arginine
S	Serine
T	Threonine
U	Selenocysteine
V	Valine
W	Tryptophan
Y	Tyrosine
Z	Glutamic acid or Glutamine
X	any
*	translation stop
-	gap of indeterminate length

Tabella B.2: Codici IUB/IUPAC per gli aminoacidi usati nel formato FASTA.

Bibliografia

[1] A. Rich. Dna comes in many forms. *Gene*, 135(1-2):99–109, Dicembre 1993. [citato a p. 6]

[2] Margaret G. Kidwell. Transposable elements. In Dr. T. Ryan Gregory, editor, *The Evolution of the Genome*, chapter 3, pages 165–221. Elsevier, San Diego, Dicembre 2004. [citato a p. 16]

[3] G. B. Singh. Discovering Matrix Attachment Regions (MARs) in genomic databases. *ACM SIGKDD Explorations Newsletter*, 1:39–45, 2000. [citato a p. 17]

[4] K. Liolios, K. Mavromatis, N. Tavernarakis, and N.C. Kyrpides. The Genomes On Line Database (GOLD) in 2007: status of genomic and metagenomic projects and their associated metadata. Database issue D475-D479, N.A.R., 36, 2008. [citato a p. 18, 22]

[5] R. J. Robbins. Challenges in the Human Genome Project. *IEEE Engineering in Medicine and Biology*, 11:25–34, 1992. [citato a p. 20]

[6] J. Craig Venter, Mark D. Adams, and Eugene W. Myers *et al.* The Sequence of the Human Genome. *Science*, 291(5507):1304–1351, Febbraio 2001. http://www.sciencemag.org/cgi/content/full/291/5507/1304. [citato a p. 22]

[7] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26:290–309, 2000. [citato a p. 22, 27]

[8] S. Heber and J. Stoye. Finding all common intervals of k permutations. *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 207–218, 2001. [citato a p. 22, 38]

[9] I.B. Rogozin, K.S. Makarova, J. Murvai, E. Czabarka, Y.I. Wolf, R.L. Tatusov, L.A. Szekely, and E.V. Koonin. Connected gene neighborhoods in prokaryotic genomes. *Nucleic Acids Res.*, 30:2212–2223, 2002. [citato a p. 22]

[10] I. Yanai and C. DeLisi. The society of genes: networks of functional links between genes from comparative genomics. *Genome Biol.*, 3(64):1–12, 2002. [citato a p. 22]

[11] A. Amir, A. Apostolico, G.M. Landau, and G. Satta. Efficient text fingerprinting via parikh mapping. *Journal of Discrete Algorithms*, 1(5):409–421(13), 2003. [citato a p. 22]

[12] A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. *In Proceedings of the ninth Annual InternationalConference on Computing and Combinatorics (COCOON)*, pages 68–79, 2003. [citato a p. 22, 32]

[13] R. Eres, L. Parida, and G.M. Landau. A combinatorial approach to automatic discovery of cluster-patterns. *Proceedings of the Third Workshop on Algorithms in Bioinformatics (WABI)*, 2812:139–150, 2003. [citato a p. 22, 30]

[14] A. Bergeron, M. Blanchette, A. Chateau, and C. Chauve. Reconstructing ancestral gene orders using conserved intervals. *Proceedings of the Fourth Workshop on Algorithms in Bioinformatics (WABI)*, pages 14–25, 2004. [citato a p. 22, 32]

[15] T. Schmidt and J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 347–358, 2004. [citato a p. 22, 55]

[16] Gad M. Landau, Laxmi Parida, and Oren Weimann. Using PQ Trees for Comparative Genomics. *Journal of Computational Biology*, 12(10):1289–1306, 2005. [citato a p. 22, 32, 35, 36, 37, 40, 48]

[17] Y. Zheng, B. P. Anton, R. J. Roberts, and S. Kasif. Phylogenetic detection of conserved gene clusters in microbial genomes. *BMC Bioinformatics*, 6:243, 2005. [citato a p. 22, 26]

[18] Zaky Adam, Monique Turmel, Claude Lemieux, and David Sankoff. Common intervals and symmetric difference in a model-free phylogenomics, with an application to streptophyte evolution. *Journal of Computational Biology*, 14(4):436–445, Maggio 2007. [citato a p. 22, 32]

[19] A. Bergeron, Y. Gingras, and C. Chauve. Formal models of gene clusters. In I. Mandoiu and A. Zelikovsky, editors, *Bioinformatics Algorithms: Techniques and Applications*, chapter 8, pages 177–202. Wiley, Marzo 2008. [citato a p. 23]

[20] T. Dobzhansky and A.T. Sturtevant. Inversions in the chromosomes of drosophila pseudoobscura. *Genetics*, 23:28–64, 1938. [citato a p. 23]

[21] Guillaume Bourque, Pavel A. Pevzner, and Glenn Tesler. Reconstructing the genomic architecture of ancestral mammals: Lessons from human, mouse, and rat genomes. *Genome Research*, 14(4):507–516, April 2004. [citato a p. 24, 52]

[22] Eugeni Belda, Andres Moya, and Francisco J. Silva. Genome Rearrangement Distances and Gene Order Phylogeny in gamma-Proteobacteria. *Mol Biol Evol*, 22(6):1456–1467, 2005. [citato a p. 24, 52]

[23] Zheng Fu, Xin Chen, Vladimir Vacic, Peng Nan, Yang Zhong, and Tao Jiang. Parsimony approach to genome-wide ortholog assignment. *RECOMB 2006, LNBI 3909*, 9:578–594, 2006. [citato a p. 24]

[24] X. She *et al.* A preliminary comparative analysis of primate segmental shows elevated substitution rates and a great-ape expansion of intrachromosomal duplications. *Genome Research*, 16(5):576–583, 2006. [citato a p. 25]

[25] M.N. Price, A.P. Arkin, and E.J. Alm. The life-cycle of operons. *PLoS Genetics*, 2(5):e96, 2006. [citato a p. 25]

[26] R. M. Karp. Mapping the genome: Some combinatorial problems arising in molecular biology. *Proc. 25th Annual ACM Symp. on Theory of Computing*, pages 278–285, 1993. [citato a p. 32]

[27] A. Bergeron, J. Mixtacki, and J. Stoye. Reversal distance without hurdles and fortresses. *In Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 388–399, 2004. [citato a p. 32]

[28] Anne Bergeron, Cedric Chauve1, Fabien de Montgolfier, , and Mathieu Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. *Lecture Notes in Computer Science*, 3669:779–790, 2005. [citato a p. 32]

[29] Kellogg S. Booth, Lueker, and George S. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976. [citato a p. 32]

[30] G. Camposeo. Scoperta di pattern ripetuti mediante l'uso di alberi PQ. Master's thesis, Università degli Studi di Pisa, Aprile 2008. [citato a p. 40]

[31] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Mol. Biol.*, 215:403–410, 1990. [citato a p. 44]

[32] P. Weiner. Linear pattern matching algorithms. *Proceedings of the IEEE 14th Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973. [citato a p. 67]

[33] E. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23:262–272, 1976. [citato a p. 67, 70]

[34] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995. [citato a p. 67]

[35] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997. [citato a p. 69]

[36] U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22:935–948, 1993. [citato a p. 70]

[37] M. Crochmore and R. Vérin. Direct construction of compact acyclic word graphs. *Symposium on Combinatorial Pattern Matching (CPM'97)*, 1264:116–129, 1997. [citato a p. 70]

[38] S. Kurtz. Reducing the space requirement of suffix trees. *Software: Practice and Experience*, 29(13):1149–1171, 1999. [citato a p. 70]

[39] P. Ferragina and G. Manzini. Indexing compressed texts. *Journal of the ACM*, 52(4):552–581, 2005. [citato a p. 70]

[40] R. Grossi and J. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. *SIAM Journal on Computing*, 35(2):378–407, 2006. [citato a p. 70]

[41] K. Sadakane. Compressed suffix trees with full functionality. *Theory of Computing Systems*, To appear, preliminary version available at `http://tcslab.csce.kyushu-u.ac.jp/~sada/papers/cst.ps`, 2007. [citato a p. 70]

[42] J. I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001. [citato a p. 70]

[43] Niko Välimäki, Wolfgang Gerlach, Kashyap Dixit, and Veli Mäkinen. Engineering a compressed suffix tree implementation. *LNCS*, 4525:217–228, 2007. [citato a p. 71]

[44] D. Laforenza, R. Lombardo, M. Scarpellini, M. Serrano, F. Silvestri, and P. Faccioli. Biological experiments on the grid: A novel workflow management platform. In *CBMS* [45], pages 489–494. [citato a p. 80]

[45] *20th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2007), 20-22 June 2007, Maribor, Slovenia*. IEEE Computer Society, 2007. [citato a p. 92]

[46] R. Dulbecco and P. Vezzoni. Il progetto genoma umano. *Le Scienze*, 100(D), marzo 1998. [citato a p. -]

[47] J. I. Munro, V. Raman, and S. Srinivasa Rao. Space efficient suffix trees. *Journal of Algorithms*, 39(2):205–222, 2001. [citato a p. -]

[48] R. Grossi, A. Gupta, and J. Vitter. High-order entropy-compressed text indexes. *14th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA '03)*, pages 841–850, 2003. [citato a p. -]

[49] P. Ferragina and G. Manzini. Opportunistic data structures with applications. *41st IEEE Symposium on Foundations of Computer Science*, pages 390–398, 2000. [citato a p. -]

[50] B. Alberts *et al*. *Molecular Biology of the Cell*. Garland Science, fifth edition edition, 2007. [citato a p. -]

[51] B. Lewin. *Genes IX*. Prentice Hall, 2007. [citato a p. -]

[52] A. La Terza, V. Passini, and S. Barchetta *et al*. Adaptive evolution of the heat-shock response in the antarctic psychrophilic ciliate, *euplotes focardii*: hints from a comparative determination of the hsp70 gene structure. *ANTARCTIC SCIENCE*, 1-2:239–244, 2004. [citato a p. -]

Tavola dei Simboli e Abbreviazioni

Abbreviazione	Descrizione	Definizione
A	Adenina	pagina 2
C	Citosina	pagina 2
T	Timina	pagina 2
G	Guinina	pagina 2
DNA	Acido Deossiribonucleico	pagina 2
bp	base pair	pagina 3
NMR	*Nuclear Magnetic Resonace*	pagina 8
U	Uracile	pagina 9
RNA	Acido Ribonucleico	pagina 9
TF	Transcription Factor	pagina 12
TFBS	Transcription Factor Binding Site	pagina 12
UTR	Untraslated Region	pagina 14
MAR	Matrix Attachment Regions	pagina 16
S/MAR	Scaffold/matrix Attachment Regions	pagina 16
ORI	Origin Of Replication	pagina 17
Σ	Insieme dei simboli, o alfabeto	pagina 24
Π	Insieme di permutazioni	pagina 27
π	Generica permutazione di caratteri	pagina 30
‘,’	Alberi PQ/Permutazioni: elementi permutati	pagina 31
‘–’	Alberi PQ/Permutazioni: elementi affiancati	pagina 31

Abbreviazione	Descrizione	Definizione
$\mathcal{G}$	Genoma come insieme di stringhe	pagina 28
$\mathcal{CS}$ o $\mathcal{F}$	Fingerprint di una stringa	pagina 30
T_U	Albero PQ universale	pagina 33
$\mathcal{C}(T)$	$\{F(T')\|T' \equiv T\}$	pagina 33
$F(T)$	Frontiera dell'albero T	pagina 33
NCBI	*National Center for Biotechnology Information*	pagina 81
PCR	*Polymerase Chain Reaction*	pagina 84

Elenco delle Figure

Elenco delle Tabelle

Elenco degli Algoritmi

Indice analitico