



UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di laurea in Informatica

Rilevazione di eventi di Alternative Splicing a partire da read paired-end

Relatore: Prof. Della Vedova Gianluca

Correlatore: Prof. Rizzi Raffaella

Relazione della prova finale di:

Francesco Porto

Matricola 816042

Anno Accademico 2018-2019

Ai miei genitori Patrizia e Rosario, alle mie nonne Maria e Bepina, a Mariarosa per...
tutto!

A Simone, Daniele, Ayoub, Mattia, Edoardo, Brian, Federico per avermi accompagnato
in questo percorso.

A Luca Denti e ai proff. Della Vedova e Rizzi, per il loro supporto durante lo stage e
nella stesura di questo documento.

Abstract - Italiano

L'Alternative Splicing è un meccanismo attraverso il quale diverse proteine sono generate a partire da uno stesso gene. Si stima che oltre il 75% dei geni umani utilizzi l' Alternative Splicing, e una piena comprensione di questo meccanismo potrebbe aiutare a far luce su diversi fenomeni biologici non ancora del tutto chiari, oltre che a migliorare la capacità di rilevazione di diverse patologie di natura genetica. Con l'avvento delle tecnologie NGS (Next Generation Sequencing), l'accesso a grandi quantità di informazioni di natura biologica è diventato sempre più facile e conveniente: in questo contesto l'informatica gioca un ruolo fondamentale nello studio dell'Alternative Splicing. Le read paired-end, prodotte da sequenziatori NGS, sono ad oggi ampiamente utilizzate negli allineatori per ottenere risultati più precisi rispetto alle read single-end; il loro utilizzo permette inoltre di rendere l'identificazione di eventi di Alternative Splicing più accurata. ASGAL (Alternative Splicing Graph ALigner) è un software sviluppato dall'Algolab in grado di rilevare eventi di Alternative Splicing a partire da read single-end. Al momento non è in grado di gestire le read paired-end. Abbiamo quindi deciso di aggiungere il supporto alle read paired-end ad ASGAL, per migliorare le sue capacità di rilevazione. In questo documento saranno evidenziate le principali modifiche apportate ad ASGAL, ponendo l'attenzione sulle differenze tra il formato single-end e quello paired-end. Sarà poi presentato un esempio del suo funzionamento, oltre che ad alcuni possibili sviluppi futuri.

Abstract - English

Alternative Splicing is a mechanism by which different proteins are produced starting from the same gene. It is estimated that over 75% of human genes undergo Alternative Splicing, and a full comprehension of such mechanism could help shed light on different biological phenomena which are not fully understood yet, and also to improve the ability to detect genetic diseases. With the advent of NGS (Next Generation Sequencing) technologies, access to biological data has become easier and cheaper: in this context computer science plays a key role in the study of Alternative Splicing. Paired-end reads, produced by NGS sequencers, are nowadays widely used by aligners in order to obtain more precise results when compared to single-end reads; their usage also allows Alternative Splicing event detection to be more accurate. ASGAL (Alternative Splicing Graph ALigner) is a software developed by Algolab capable of detecting Alternative Splicing events starting from single-end reads. Currently, it is not capable of handling paired-end reads. Therefore, we have decided to add paired-end support to ASGAL, in order to improve its event detection capabilities. In this paper the main changes to ASGAL will be highlighted, focusing our attention on differences between single-end and paired-end formats. An example of its execution will be presented, as well as some future prospects.

Contents

1	Introduzione	1
1.1	ASGAL	1
1.2	Alternative Splicing	3
1.3	Read paired-end	4
1.4	Obiettivo stage e risultati ottenuti	5
2	Generazione dello Splicing Graph e allineamento Splice-Aware	6
2.1	Descrizione generale	6
2.2	Allineamento di entrambe le read	7
2.3	Introduzione di read unmapped e placeholder nel formato mem	7
2.4	Supporto alle fragment library types	9
3	Computazione degli allineamenti Spliced	11
3.1	Descrizione generale	11
3.2	Campo FLAG per read paired-end	12
3.3	Campi RNEXT, PNEXT e TLEN	13
3.4	Calcolo delle statistiche dell'allineamento	14
4	Rilevazione degli eventi di Alternative Splicing	15
4.1	Descrizione generale	15
4.2	Merge degli introni dedotti dai due sample	16
4.3	Computazione dell' IDMP	17
4.4	Computazione del TIDMP	18
5	Sviluppi futuri	20
5.1	Incremento della qualità degli allineamenti usando IDMP e TIDMP	20
6	Esempio di funzionamento	21
6.1	Generazione delle read con Flux Simulator	21
6.2	Utilizzo di ASGAL	22
6.3	Risultati	23
7	Competenze acquisite	25
8	Conclusioni	26

1 Introduzione

1.1 ASGAL

ASGAL (Alternative Splicing Graph ALigner) [1] è un tool sviluppato dall'Algolab per l'identificazione di eventi di Alternative Splicing espressi in un campione di RNA-Seq a partire da un genoma di riferimento e dall'annotazione di un gene. ASGAL si compone di quattro step:

1. **Costruzione dello splicing graph:** a partire dall'annotazione di un gene e dalla genomica di riferimento, ASGAL costruisce uno splicing graph, ovvero una struttura a grafo che rappresenta tutti i trascritti noti del gene in input. Viene inoltre prodotta una linearizzazione dello splicing graph, che sarà utilizzata in fase di allineamento.
2. **Allineamento Splice-Aware:** utilizzando gli algoritmi in [2] e [3] ASGAL allinea le RNA-Seq reads con la linearizzazione dello splicing graph del gene in input. L'allineamento è Splice-Aware in quanto è necessario tenere traccia della posizione di esoni ed introni per un corretto allineamento.
3. **Computazione degli allineamenti Spliced:** gli allineamenti prodotti dall'Allineatore Splice-Aware vengono convertiti nel formato SAM, per permetterne l'elaborazione con strumenti standard.
4. **Rilevazione degli eventi di Alternative Splicing:** gli allineamenti prodotti dall'Allineatore Splice-Aware sono analizzati per rilevare gli eventi di Alternative Splicing indotti dalle read del campione. Gli eventi rilevati sono visualizzati in un file csv, che include diverse informazioni su ciascun evento rilevato.

ASGAL è stato sviluppato in C++ e Python: C++ viene utilizzato nella parte di allineamento per la sua efficienza, Python nella parte di rilevazione per la sua semplicità nell'utilizzo di strutture dati complesse. Il progetto è open source, ed è disponibile su GitHub con licenza GNU v3.0. Tutti i componenti sono utilizzabili singolarmente, ma viene fornito uno script principale che ne semplifica l'utilizzo.

Al momento ASGAL non supporta le read paired-end, un nuovo formato di read prodotte da allineatori NGS (Next Generation Sequencing), che potrebbe portare ad un incremento di efficacia ed efficienza nella rilevazione di eventi di Alternative Splicing.

L'immagine nella pagina successiva riassume il funzionamento di ASGAL.

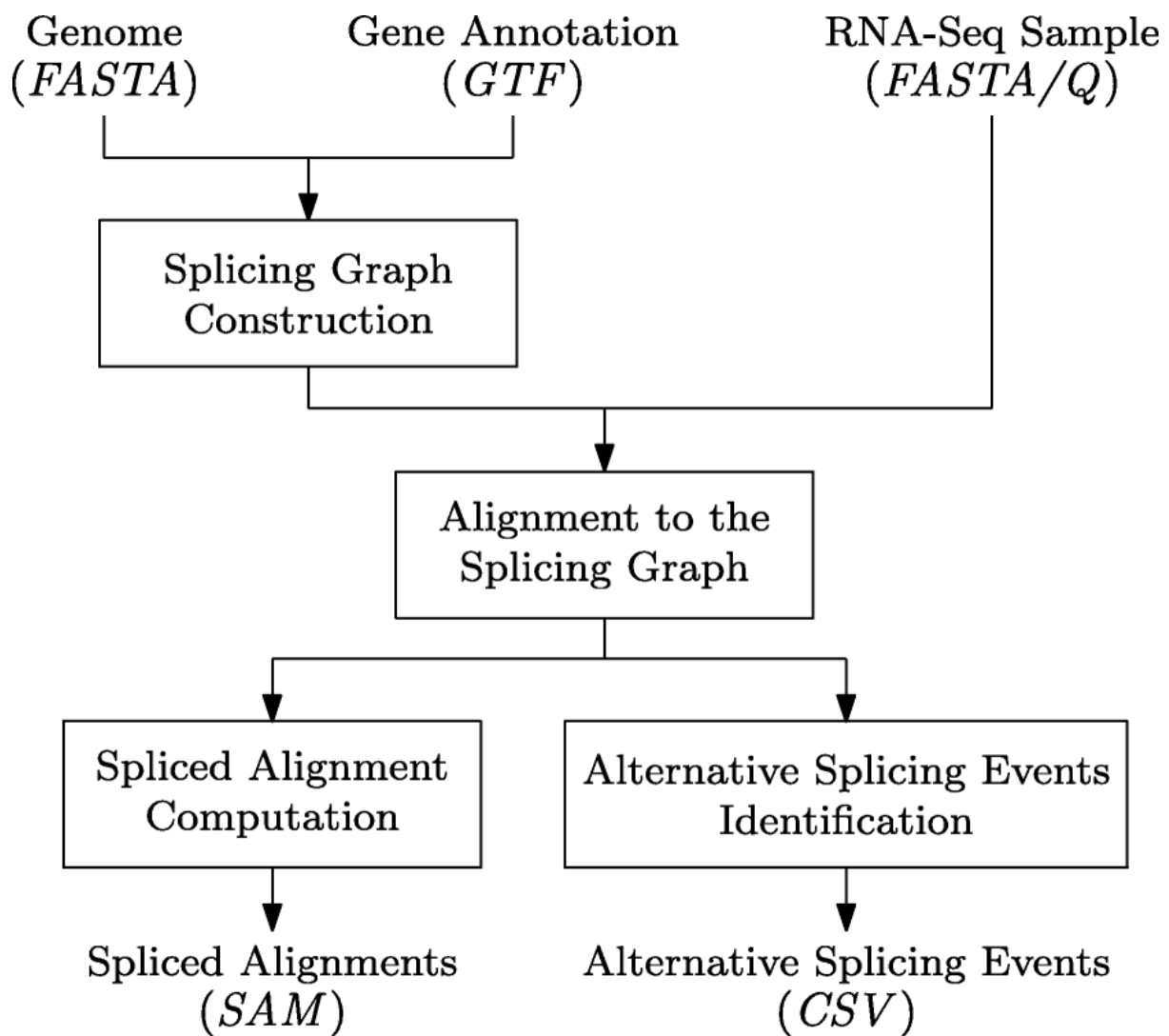


Figure 1: Il funzionamento di ASGAL illustrato

1.2 Alternative Splicing

L'alternative splicing è un meccanismo utilizzato dalle cellule per produrre proteine diverse dallo stesso gene che viene utilizzato da oltre il 75% dei geni umani [4]. Diversi studi (come [5] e [6]) dimostrano come l'alternative splicing giochi un ruolo fondamentale nello sviluppo di diverse malattie, come ad esempio il cancro o la sindrome di Alzheimer.

Considerando un generico frammento di DNA, esso può essere diviso in esoni (parti codificanti) e introni (parti non codificanti). Durante la fase di Trascrizione gli introni vengono rimossi e la Timina viene trasformata in Uracile, ottenendo pre-mRNA. A questo punto, in un normale processo di Splicing, tutti gli esoni vengono utilizzati, nell'ordine in cui appaiono nel pre-RNA, per ottenere una proteina.

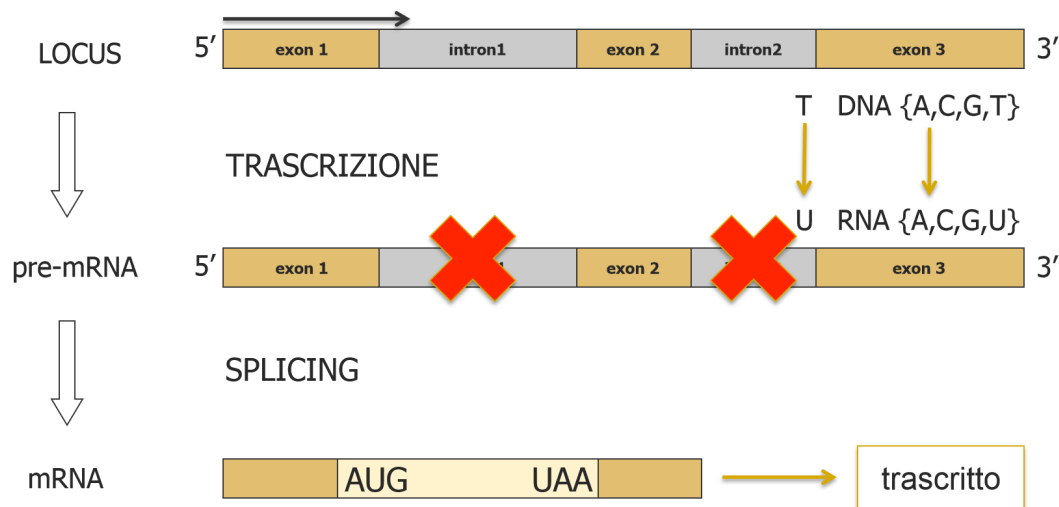


Figure 2: Trascrizione e Splicing

Nel caso di un evento di Alternative Splicing, questo non accade: alcuni esoni potrebbero infatti non essere utilizzati, o apparire in un ordine diverso. Vengono riconosciuti 5 tipi di eventi di Alternative Splicing:

1. **Exon Skipping:** Un esone non appare nel trascritto; quando gli esoni sono più di uno, si parla di **Multiple Exon Skipping**
2. **Mutually Exclusive Exons:** Due esoni non compaiono mai in uno stesso trascritto contemporaneamente
3. **Alternative 5' Donor Site:** Parte di un introne nel 5' diventa un esone
4. **Alternative 3' Acceptor Site:** Parte di un introne nel 3' diventa un esone
5. **Intron Retention:** Parte di un esone diventa un introne

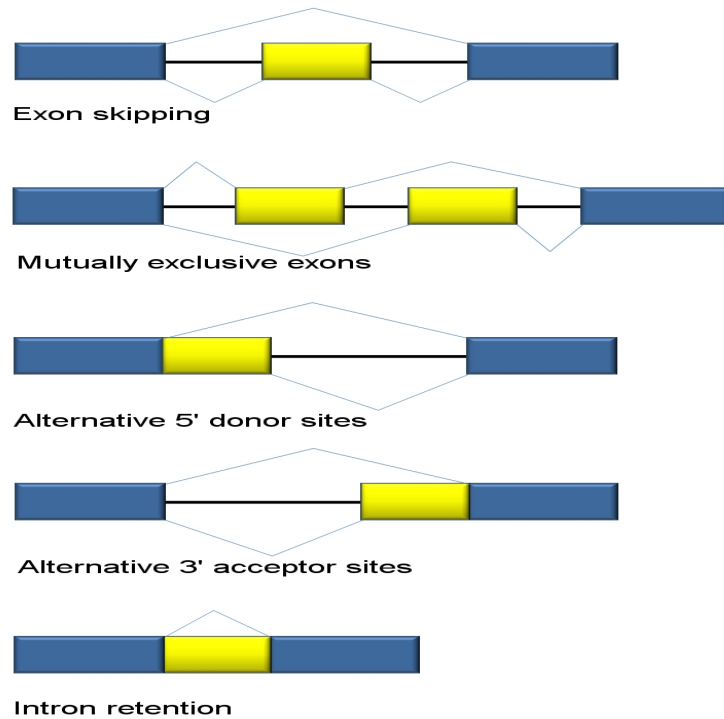


Figure 3: I diversi tipi di Alternative Splicing

1.3 Read paired-end

Le read paired-end consistono nell'estrazione di due letture da un singolo frammento di DNA o RNA, contrariamente alle single-end reads che ne estraggono solo una. Sono prodotte da sistemi NGS, e la loro preparazione è molto semplice: una volta stabilita la grandezza della singola lettura, viene estratta la lettura sull'estremità sinistra, il campione viene girato, e viene estratta nuovamente l'estremità sinistra (ottenendo quindi l'estremità destra); viene generalmente fornita anche la distanza tra le due letture.

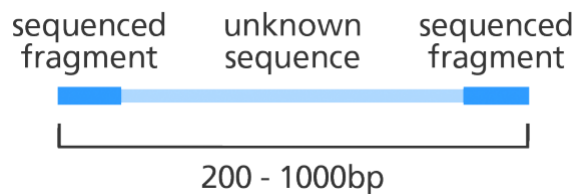


Figure 4: Read paired-end

1.4 Obiettivo stage e risultati ottenuti

Prima dell'inizio dello stage, ASGAL era in grado di gestire solo read single-end. L'obiettivo dello stage era quello di estendere ASGAL per supportare le read paired-end, ottenendo così una maggiore efficacia in fase di rilevazione.

Per ottenere questo risultato, è stato necessario modificare ognuno degli step di ASGAL:

- **Allineamento Splice-Aware:** è stata modificata la procedura di allineamento per allineare due read alla volta anziché una, sono stati introdotti nuovi tipi di MEM relativi alle read unmapped e "placeholder", è stato aggiunto il supporto alle Fragment Library Types per aumentare le prestazioni dell'allineamento
- **Computazione degli eventi Spliced:** è stato modificato come alcuni campi (FLAG, RNEXT, PNEXT e TLEN) del formato SAM vengono computati, vengono ora visualizzate alcune statistiche sull'allineamento
- **Rilevazione di eventi di Alternative Splicing:** la procedura di rilevazione è ora in grado di utilizzare gli introni dedotti da entrambe le read; sono state calcolate le statistiche IDMP e TIDMP che potrebbero, in futuro, essere usate per migliorare la qualità e quantità degli eventi rilevati.

ASGAL è ora in grado di allineare correttamente read paired-end con un genoma di riferimento, di salvare gli allineamenti ottenuti nel formato SAM (rispettando le specifiche per quanto riguarda read paired-end) e di visualizzare gli eventi di Alternative Splicing rilevati.

Nelle pagine successive saranno evidenziate nel dettaglio tutte le modifiche apportate ad ASGAL.

2 Generazione dello Splicing Graph e allineamento Splice-Aware

2.1 Descrizione generale

Il primo compito di ASGAL è quello di generare lo Splicing Graph a partire dal genoma di riferimento (in formato FASTA¹) e dalla sua annotazione (in formato GTF²). Uno splicing graph è un grafo orientato in cui i nodi rappresentano gli esoni, e gli archi collegano due esoni che compaiono uno dopo l'altro in un trascritto; utilizzando la chiusura transitiva ogni esone viene poi collegato a tutti i suoi successori.

Per permettere un allineamento tra due stringhe viene generata una linearizzazione dello Splicing Graph, ottenuta semplicemente concatenandone i nodi e utilizzando un carattere speciale come delimitatore tra un nodo e un altro.

Questa prima fase non utilizza le read paired-end, e non è stata quindi modificata nell'adattamento al nuovo tipo di read.

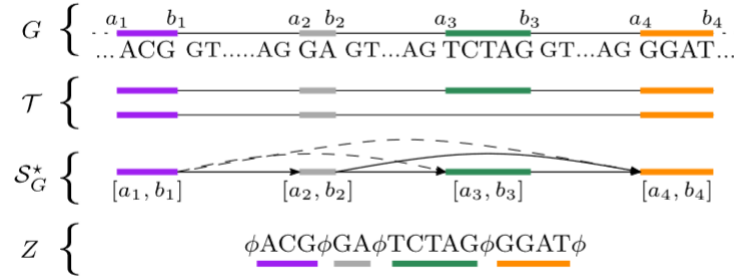


Figure 5: Un esempio di Splicing Graph

Il secondo compito di ASGAL è quello di allineare le read paired-end con la linearizzazione appena ottenuta. Per fare questo, utilizza il concetto di MEM (Maximum Exact Matches), ovvero di una tripla così costituita:

1. Posizione iniziale nel genoma
2. Posizione iniziale nella read
3. Lunghezza della parte in comune

Come si può intuire dal nome, i MEM utilizzano il concetto di pattern matching esatto, rispetto ai generici algoritmi di allineamento che ammettono dei mismatch o degli indel; ad ogni allineamento corrispondono uno o più MEM.

Il nuovo formato paired-end è stato qui utilizzato per velocizzare il processo di allineamento ove possibile. Al momento non sono state utilizzate per migliorare l'efficacia dell'allineamento; si rimanda alla sezione "Sviluppi futuri" per informazioni su come questo potrebbe accadere.

¹https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp

²<https://www.ensembl.org/info/website/upload/gff.html>

Il risultato dell'allineamento sarà restituito, per comodità, nel formato *.mem* così composto:

- Lo **strand** dell'allineamento
- Il **nome della read** allineata
- Il **numero di errori** commessi nell'allineamento
- Il **MEM** vero e proprio, eventualmente anche più di uno
- La **sottostringa della read** che costituisce il MEM considerato

2.2 Allineamento di entrambe le read

La prima differenza sostanziale rispetto al caso single-end è che nel caso paired-end le read da allineare sono sostanzialmente due (ovvero le due estremità); bisogna inoltre tenere conto della distanza tra di esse.

Mentre in un generico allineatore questo avrebbe creato non pochi problemi, in ASGAL la soluzione è banale: basta trattare le due estremità come read singole, e allinearle indipendentemente. Questo è possibile in quanto ciò che si vuole ottenere è un insieme di MEM, e non siamo interessati all'ottimizzazione globale dell'allineamento.

Resta comunque fondamentale riuscire a collegare i MEM delle due estremità tra di loro. Per fare questo, abbiamo deciso di utilizzare due file *.mem* anziché uno, con la seguente proprietà: una stessa riga nel primo file e nel secondo corrispondono alla stessa read. Questo risolve il problema del collegamento tra MEM provenienti dalla stessa read, e indirettamente anche quello della distanza tra le estremità (dati i MEM è possibile calcolare la distanza).

Non è però detto che entrambe le estremità siano coinvolte nello stesso numero di allineamenti (potrebbero addirittura non esserlo affatto). E' quindi necessario introdurre nuovi tipi di MEM per questi casi.

2.3 Introduzione di read unmapped e placeholder nel formato mem

Nei file *mem* ottenuti dallo Splice-Aware Aligner vengono ora visualizzati due nuovi tipi di MEM: quelli relativi alle read unmapped e quelli relativi ai placeholder. Il primo caso è banale, e rappresenta tutte quelle read che non hanno un matching esatto di lunghezza considerevole con il genoma dato in input.

Il secondo caso è più complesso e rappresenta un insieme di read fasulle utilizzate solo come padding per avere due file MEM della stessa lunghezza: questo facilita enormemente l'elaborazione nello step successivo (la formattazione SAM). Come detto in precedenza quando si lavora con read paired-end è sempre necessario lavorare a coppie, ma non sempre ad uno stesso pair è associato lo stesso numero di allineamenti secondari: è qui che entrano in gioco i placeholder.

I placeholder sono stati implementati in questo modo: si tengono due contatori (che rappresentano rispettivamente il numero di allineamenti relativi alla prima read e quelli relativi alla seconda read), si ottengono separatamente gli allineamenti relativi a ciascuna delle due read, e si controllano i contatori. Si prende il minore dei due e si aggiungono tanti placeholder quanto bastano per rendere uguali i contatori.

Il seguente frammento di codice mostra la procedura:

```
//head_1 and head_2 are the IDs of the alignments
if (count1 < count2) {
    while (count1 < count2) {
        outFile_1 << "PLACEHOLDER" << "_" << head_1 << "\n";
        count1++;
    }
} else if (count2 < count1) {
    while (count2 < count1) {
        outFile_2 << "PLACEHOLDER" << "_" << head_2 << "\n";
        count2++;
    }
}
```

E' stato inoltre necessario introdurre un nuovo campo nel formato *.mem*, che permette di distinguere il tipo di MEM preso in esame: ad esempio trovando come primo valore la stringa "MAPPED", si capisce che il MEM preso in esame è relativo ad una read che è stata mappata correttamente; ovviamente vale lo stesso ragionamento con le stringhe "UNMAPPED" e "PLACEHOLDER".

L'immagine seguente mostra un esempio di file nel nuovo formato mem:

```
MAPPED + 21:6630182-6638337C:ENST00000623047:2:5990:1011:1176/2 0 (2559,1,100)
TTCACCTGGGTCAGTGTCTGTAGCATGAACAGTCCTCCACCAACCTGTACAGTCTCTCACTTCTGGTTTCTTAATAGCACCTTCCCCTCTTTACCT
UNMAPPED 21:6630182-6670695C:ENST00000625185:3:5990:28:26/2
GTGCCAGCACCAGCGAGAGACAGATCCCGCCAAATGCAGGAAGCCACGCCCTC
MAPPED + 21:6630182-6638337C:ENST00000623047:4:5990:4282:4484/2 0 (5242,1,100)
TAAAGAGTGCATGTTTGCATAACAATCCTAAATTAATATTTTAGAATAATAGCAATGTTTGTGTTTCAAGTGGGCGGTGTTCACTCAGGACATCA
MAPPED + 21:6630182-6638337C:ENST00000623047:5:5990:5443:5689/2 0 (4037,1,100)
TTGATTGACTGATTAAGTGTCTGAGGAGGAACATATGTAGGGAACAGCCTGGGTCTTTGAATCCCTGTTCCCGAGCTATGATGCCTGTGCAAATG
MAPPED + 21:6630182-6638337C:ENST00000623047:6:5990:3623:3776/2 0 (5950,1,43) (8956,43,58)
CATCGCATATCTGGAGTTCGGGGTCTTAGAAAGCTTTCTTGCCCTATTTCTTTAGCAGAATGAGTGTGCTACATTTCCAGGACTGTTTTTATTTGTCT
PLACEHOLDER 21:6630182-6638337C:ENST00000623047:7:5990:2613:2804/2
MAPPED + 21:6630182-6638337C:ENST00000623047:8:5990:2106:2288/2 0 (1447,1,100)
AGAAGGGGAACAAGCAAAGTTTGTAAACAAGTACTCTGTTTGGACCTGAAGACTAAATTACAGAATGGTTGTTTCATTTTTAAAAATAGGAATTTGTA
MAPPED + 21:6630182-6638337C:ENST00000623047:9:5990:540:744/2 0 (2991,1,100)
GGGTTTGAGCAGTCTCTGCATCAACCGCTTAATTACCTAGGACTATAGGCATGAACCACCATGCCTGGCTAGCTTTATTTATTTTGTGTTTTATTTT
MAPPED + 21:6630182-6638337C:ENST00000623047:10:5990:2910:3147/2 0 (588,1,100)
TGCCTCGGCCTCCCAAGTAAGTGGGATTACAGGAATACACGACCACCCCGTTTAAAGTTTGTATTTTAGTAGAGACTGTGTTTCTTCATGTTAGTGAGG
MAPPED + 21:6630182-6638337C:ENST00000623047:11:5990:2778:2918/2 0 (817,1,100)
GCTGGATTCTTCAACATGAAGTATTTTTAAATTGAAACTAATTGAATGACTTTAACTGGTAAGTAGAAGTCTTAGACCGTTGACTAAAAGCTAAGGC
```

Figure 6: Esempio di file in formato *.mem*

2.4 Supporto alle fragment library types

Ci sono diversi protocolli per la preparazione di librerie paired-end, che portano a read con caratteristiche diverse. Le **fragment library types**³ permettono di descrivere queste caratteristiche in modo sintetico. Le caratteristiche che possono essere descritte sono:

- **Orientamento relativo di una read rispetto all'altra:** può essere inward (I) o outward (O)
- **Se è noto o meno lo strand di appartenenza delle due read:** può essere stranded (S) o unstranded (U)
- **La direzionalità della prima read, solo nel caso stranded:** può essere first (F) o reverse (R)

La seguente immagine descrive tutte le possibili combinazioni:

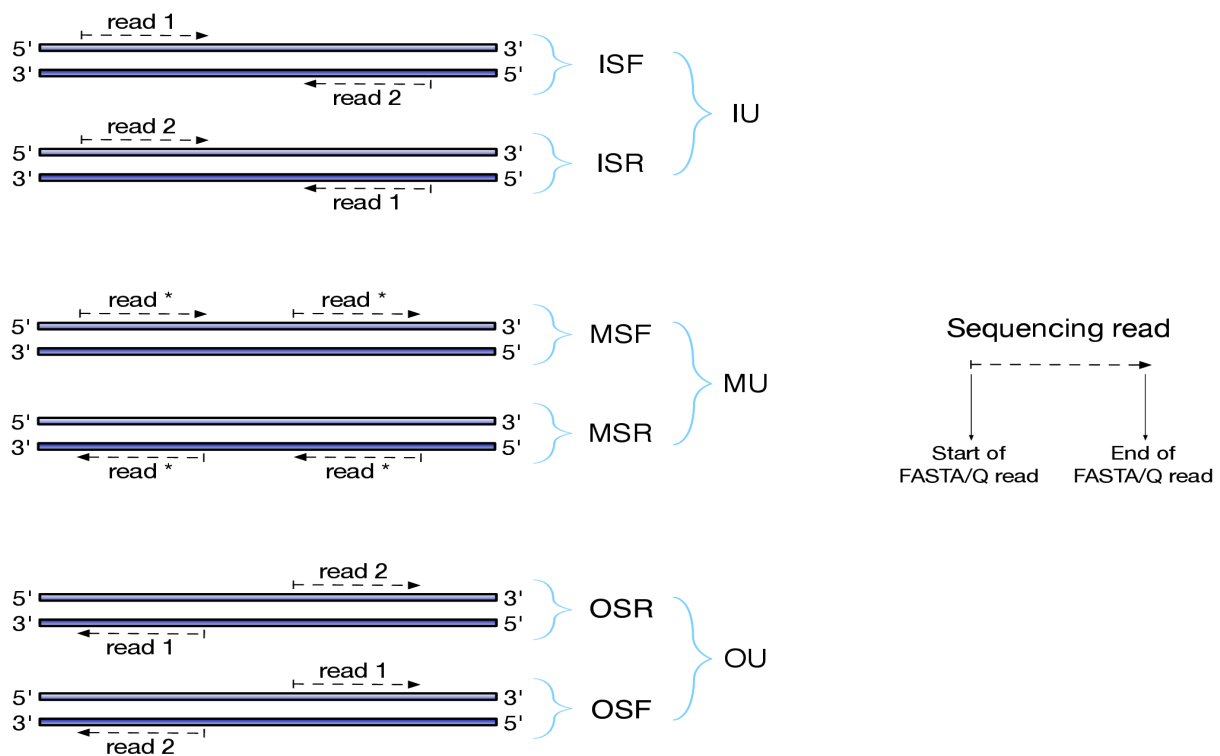


Figure 7: I diversi tipi di FTL

³<https://salmon.readthedocs.io/en/latest/>

ASGAL utilizza queste informazioni per velocizzare il processo di allineamento. Nel formato single-end questa informazioni non esiste, quindi ogni read veniva allineata in entrambe le direzioni, e si prendeva l'allineamento migliore dei due.

Nel formato paired-end, si può utilizzare la ftl per ridurre il numero di allineamenti necessari, fino al 50% nel caso stranded. Ad esempio, se viene fornita un ftl di tipo ISF è sufficiente allineare la read 1 sullo strand + (ignorando lo strand -) e la read 2 sullo strand - (ignorando lo strand +).

Il caso unstranded richiede un po' più di attenzione: non è infatti possibile sapere a priori su quale strand allineare la prima read. Per risolvere questo problema è stata riutilizzata la vecchia procedura di allineamento della prima read in entrambe le direzioni; una volta trovata la direzione migliore per la prima, la seconda viene di conseguenza.

Supponiamo ad esempio di avere una libreria in formato MU: se la prima read allinea sullo strand +, di conseguenza anche la seconda sarà allineata sullo strand +; viceversa, se la prima read allinea sullo strand -, anche la seconda allinea sempre sullo strand -. In questo caso si ottiene solo una riduzione del 25% nel numero degli allineamenti.

Qualora il tipo di libreria non venga fornito dall'utente, rimane necessario provare ad allineare le read in entrambe le direzioni, senza alcun incremento di prestazioni.

3 Computazione degli allineamenti Spliced

3.1 Descrizione generale

Come già detto, gli allineamenti ottenuti dallo Splice-Aware aligner sono in un formato non standard chiamato mem, che contiene solo informazioni relative ai MEM ottenuti in fase di allineamento. L'obiettivo di questa seconda parte è quello di convertire i due file mem in un singolo file SAM⁴ (Sequence Alignment Map), il formato standard per memorizzare gli allineamenti.

Il formato SAM è composto da 11 campi:

1. **QNAME**: Il nome identificativo dell'allineamento
2. **FLAG**: Una serie di flag binari che identificano le caratteristiche dell'allineamento
3. **RNAME**: Identificativo del gene di riferimento
4. **POS**: Posizione 1-based di inizio dell'allineamento sul genoma
5. **MAPQ**: Valore che indica la qualità dell'allineamento
6. **CIGAR**: Stringa che identifica le operazioni effettuate per ottenere l'allineamento
7. **RNEXT**: QNAME del mate (solo paired-end)
8. **PNEXT**: POS del mate (solo paired-end)
9. **TLEN**: Distanza tra mate (solo paired-end)
10. **SEQ**: Allineamento vero e proprio
11. **QUAL**: Valore che indica la qualità delle read

Non si tratta solo di una semplice conversione, in quanto è necessario indurre diverse informazioni aggiuntive per avere un file SAM standard, quali: la posizione di inizio dell'allineamento sulla genomica, la stringa CIGAR, i flag relativi all'allineamento, ecc.

Per supportare le read paired-end è stato necessario modificare gran parte di queste funzionalità. In particolare, è stata modificato il modo in cui vengono calcolati i campi FLAG, RNEXT, PNEXT e TLEN.

⁴<http://samtools.github.io/hts-specs/SAMv1.pdf>

3.2 Campo FLAG per read paired-end

Il campo FLAG è il secondo del formato SAM e consiste di un valore numerico (ottenuto convertendo in decimale una serie di flag binari) che rappresenta le caratteristiche dell'allineamento preso in esame. La seguente immagine mostra il significato di ciascun bit del flag:

Binary (Decimal)	Hex	Description
00000000001 (1)	0x1	Is the read paired?
00000000010 (2)	0x2	Are both reads in a pair mapped “properly” (i.e., in the correct orientation with respect to one another)?
00000000100 (4)	0x4	Is the read itself unmapped?
00000001000 (8)	0x8	Is the mate read unmapped?
00000010000 (16)	0x10	Has the read been mapped to the reverse strand?
00000100000 (32)	0x20	Has the mate read been mapped to the reverse strand?
00001000000 (64)	0x40	Is the read the first read in a pair?
00010000000 (128)	0x80	Is the read the second read in a pair?
00100000000 (256)	0x100	Is the alignment not primary? (A read with split matches may have multiple primary alignment records.)
01000000000 (512)	0x200	Does the read fail platform/vendor quality checks?
10000000000 (1024)	0x400	Is the read a PCR or optical duplicate?

Figure 8: Il significato di ciascun bit del campo FLAG

Nei casi single-end solo due flag vengono utilizzati: quello relativo allo strand (0x16) e quello relativo al tipo di allineamento (0x100); visto che le read non sono paired, il flag 0x1 sarà sempre false, quindi tutti i flag risultanti saranno pari.

Nei casi paired-end (quasi) tutti i flag vengono utilizzati. E' inoltre necessario trattare gli allineamenti a coppie, in quanto il campo FLAG esprime informazioni anche sul mate e non solo sull'allineamento preso in esame.

Supponiamo ad esempio di avere due read, la prima che mappa sullo strand positivo e la seconda che non mappa (ed è quindi *unmapped*). Sarà innanzitutto necessario mettere a true il flag relativo alle read paired-end (0x1) per entrambe le read. Considerando la prima, sarà messo a true il flag relativo al mate unmapped (0x8) e il flag relativo al first-in-pair (0x4). Considerando la seconda, sarà messo a true il flag relativo alla read unmapped (0x4) e quello relativo al second-in-pair (0x80). I flag in decimale saranno quindi 73 e 133.

Si noti che per il momento non viene tenuto conto dei flag 0x200 e 0x400, ma questo non è di alcuna rilevanza al fine di identificare eventi di Alternative Splicing.

3.3 Campi RNEXT, PNEXT e TLEN

I campi TLEN, RNEXT e PNEXT rappresentano rispettivamente il settimo, l'ottavo e il nono campo di ogni record del formato SAM; essi sono praticamente inutilizzati quando si allineano read single-end, ma nel formato paired-end assumono maggiore importanza. In particolare i campi RNEXT e PNEXT sono utilizzati da strumenti di visualizzazione degli allineamenti (come ad esempio IGV [7]) per permettere una corretta visualizzazione di una read e del suo mate.

Il campo RNEXT contiene il nome dell'allineamento relativo al mate (ovvero il suo campo PNAME). Per semplicità, quando i due allineamenti sono consecutivi, si può lasciare il suo valore a '='.

Il campo PNEXT contiene la posizione iniziale 1-based dell'allineamento relativo al mate (ovvero il suo campo POS. Qualora il mate fosse unmapped, si utilizza il valore 0.

Il campo TLEN rappresenta la distanza la lunghezza del template osservato, ovvero la distanza (sul genoma) tra l'inizio della prima read e la fine della seconda. Per la sua computazione è sufficiente trovare la posizione finale del secondo allineamento e sottrarre la posizione iniziale del primo.

La seguente immagine mostra un esempio di allineamento visualizzato da IGV:

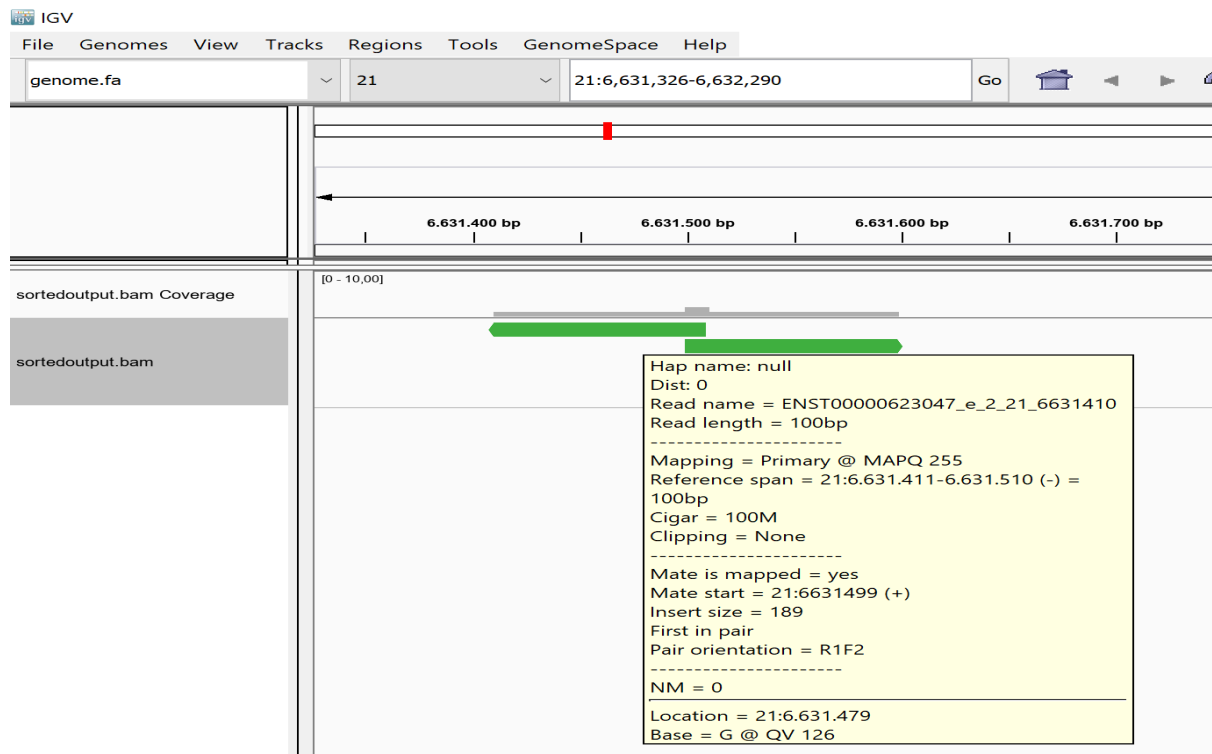


Figure 9: Informazioni sull'allineamento di una read e del suo mate usando IGV

E' importante notare che, se questi campi sono settati correttamente, lasciando il cursore del mouse su un allineamento vengono visualizzate tutte le informazioni relative al mate. Al contrario, se si prova ad indicizzare il file BAM (la versione binaria del formato SAM) per l'utilizzo con IGV, e questi campi non sono stati settati correttamente, verrà visualizzato un errore. Un esempio errore è il seguente: se il campo FLAG non contiene 0x8 (quindi il mate è mapped), e si inserisce un valore di PNEXT diverso da 0, al momento dell'indicizzazione sarà visualizzato il messaggio "mapped mate cannot have zero coordinate; treated as unmapped".

3.4 Calcolo delle statistiche dell'allineamento

Seguendo l'esempio di altri allineatori (come ad esempio STAR), è stato deciso di visualizzare alcune statistiche in fase di allineamento, quali:

- Numero di MEM mappati, non mappati e "placeholder"
- Numero di allineamenti primari e secondari
- Numero di allineamenti in cui solo una delle due read è stata mappata

Questi valori sono visualizzati nel file *.alignsinfo.txt*. Anche se non hanno finalità particolari per la rilevazione di eventi di Alternative Splicing, essi forniscono uno strumento per valutare la qualità degli allineamenti effettuati da ASGAL.

```
Count mapped1: 1281/1313
Count mapped2: 1291/1295
Count unmapped1: 32
Count unmapped2: 4
Count placeholders1: 69
Count placeholders2: 87
Count mapped pairs: 1194
Count primary alignments: 958
Count secondary alignments: 236
Count one-side alignments: 184
```

Figure 10: Esempio di file *.alignsinfo.txt*

4 Rilevazione degli eventi di Alternative Splicing

4.1 Descrizione generale

Il Rilevatore di Eventi di Alternative Splicing si occupa di rilevare tutti gli eventi indotti dalle read. Per fare questo, confronta gli introni noti (deducibili dal file gtf) con quelli *novel* (deducibili dagli allineamenti in formato mem); analizzando le differenze tra i due è possibile rilevare gli eventi di Alternative Splicing.

Trovare gli introni noti è semplice, in quanto nel file gtf sono riportati tutti gli esoni con le relative posizioni di inizio e fine: gli introni sono quindi rappresentati dalla porzione di genoma tra un esone e un altro.

La rilevazione degli introni novel è invece più complessa, e utilizza i MEM generati dallo Splice-Aware Aligner. Preso un generico allineamento, esso può essere rappresentato da uno o più MEM. Nel secondo caso viene calcolata la distanza tra di essi sia sul genoma che sul testo degli esoni; si considerano introni tutte le porzioni del genoma dove questi due valori differiscono.

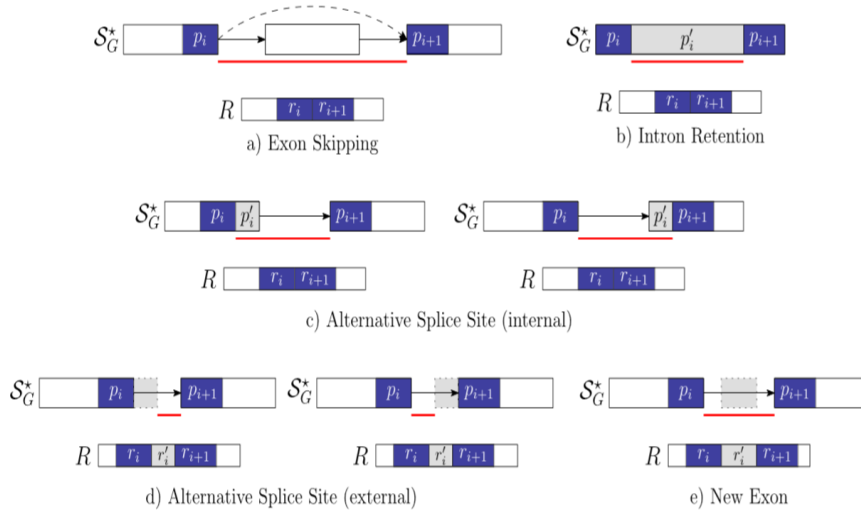


Figure 11: Fase di confronto degli introni

Viene poi eseguita una fase di riconciliazione (che permette di migliorare la qualità degli introni rilevati) e vengono filtrati quelli non supportati da un numero sufficiente di read.

A questo punto avviene la procedura di rilevazione vera e propria, che confronta gli introni novel con quelli noti, utilizzando dei pattern che rappresentano i vari eventi di Alternative Splicing.

La seguente immagine riassume la procedura:

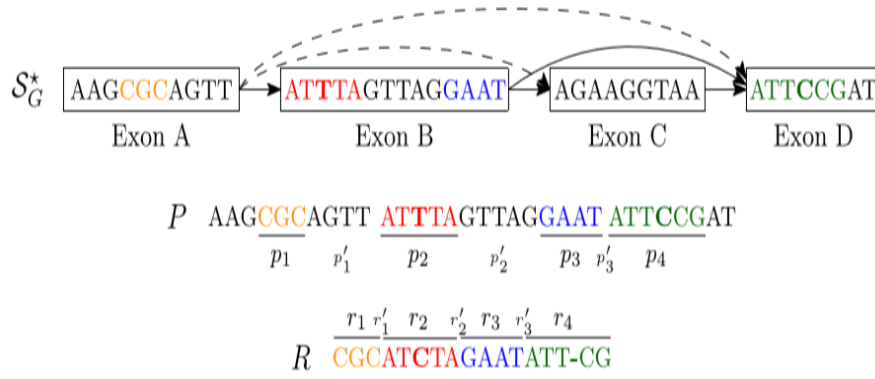


Figure 12: Riassunto della fase di rilevazione

Gli eventi di Alternative Splicing così rilevati saranno poi riportati in un file *.events.csv*, che contiene, per ogni evento rilevato:

1. Il **tipo di evento** rilevato
2. Le **posizioni iniziali e finali** sul genoma
3. Il **numero di read** che supportano l'evento
4. I **trascritti** che supportano l'evento

4.2 Merge degli introni dedotti dai due sample

Come già detto, nella nuova versione di ASGAL lo Splice-Aware Aligner produce due file *.mem* anziché uno, ed ognuno di essi contiene MEM che potrebbero rappresentare nuovi introni. A livello implementativo, un introne è rappresentato da un dizionario avente come chiave la coppia (posizione iniziale introne, posizione finale introne) e come valore il numero di read che lo supportano.

Nella fase di rilevazione di eventi, è ora necessario:

1. Rilevare gli introni da ciascuno dei due file *.mem*
2. Effettuare una merge degli introni rilevati

Per quanto riguarda il punto 1, è stata riutilizzata la procedura per l'estrazione degli introni attuale; quest'ultima viene applicata separatamente ai due file *.mem* per ottenere due dizionari, uno per ogni file.

Per quanto riguarda il punto 2, è stata creata una nuova procedura che fonde i due dizionari. Ci sono 3 casi:

- Introne che compare in uno solo dei dizionari: è sufficiente inserirlo nel nuovo dizionario risultante
- Introne che compare in entrambi i dizionari: è sufficiente inserire nel nuovo dizionario una nuova entry con la stessa chiave; il valore è dato dalla somma dei due valori
- Introni con posizioni iniziali e/o finali "simili": questo caso non è stato implementato in quanto avrebbe potuto portare ad un peggioramento della qualità della rilevazione

Il seguente frammento di codice mostra la procedura:

```
def mergeIntrons(introns1, introns2):
    introns = {}
    for (p1,p2),w in introns1.items():
        introns[(p1,p2)] = w
    for (p1,p2),w in introns2.items():
        if (p1,p2) not in introns.keys():
            introns[(p1,p2)] = w
        else:
            introns[(p1,p2)] += w
    return introns
```

4.3 Computazione dell' IDMP

Considerando una coppia di read, si definisce IDMP (Inner Distance between Mate Pairs) la distanza *sul genoma di riferimento* tra l'ultima base della prima read e la prima della seconda. Questa informazione viene generalmente fornita dall'ente che ha effettuato il sequenziamento, e può essere confrontata con l'IDMP rilevato durante l'allineamento per rilevare nuovi eventi di Alternative Splicing.

Visto che un allineamento può essere rappresentato da più di un MEM, non è possibile semplicemente aggiungere la lunghezza dell'allineamento alla sua posizione iniziale. Prima di poter calcolare l'IDMP è quindi necessario introdurre il concetto di BitVector, ovvero una sequenza di bit che rappresenta la posizione degli esoni nello Splicing Graph. Un BitVector è dotato di due operazioni, entrambe svolte in tempo costante:

- Rank: data una posizione, ritorna l'identificativo dell'esone di provenienza
- Select: dato l'identificativo di un esone, ritorna la posizione di partenza

Queste due operazioni permettono di calcolare l'IDMP in maniera efficace. Innanzitutto si prende l'ultimo MEM relativo all'allineamento della prima read, e si utilizza l'operazione di Rank per trovare l'esone di appartenenza. A questo punto, si utilizza l'operazione di Rank per trovare la posizione iniziale dell'esone. L'offset sarà quindi dato dalla differenza tra il MEM e la posizione iniziale dell'esone. Basta quindi aggiungere questo offset alla posizione iniziale per trovare la fine del primo allineamento.

L'inizio del secondo allineamento viene calcolato facendo prima la Rank sul primo MEM, e utilizzando la Select sul risultato ottenuto. A questo punto basterà sottrarre ad esso la fine di quello precedente, in modo da ottenere l'IDMP.

Il seguente frammento di codice riassume la procedura:

```
def getEnd(mem, bv, exPos):
    exonN = bv.rank(mem[0])

    # get starting position (on reference)
    exonStartingPos = exPos[exonN-1][0]

    # get offset from bitvector
    exonStartingPos_onT = bv.select(exonN)
    offset = mem[0] - exonStartingPos_onT + 1;

    # find the end
    end = exonStartingPos + offset + mem[2]
    return end

def getIdmp(start2, mems1, bv, exPos):
    lastMem1 = mems1[-1]

    # get ending position (on reference)
    end1 = getEnd(lastMem1, bv, exPos)

    return start2 - end1
```

4.4 Computazione del TIDMP

Per TIDMP si intende la misura della distanza *sui trascritti* tra le due read. Per calcolarlo è innanzitutto necessario ottenere l'ultimo MEM relativo alla prima read e il primo MEM relativo alla seconda. Da ciascuno di essi è possibile ottenere la posizione iniziale e la lunghezza sul bitvector.

A questo punto, usando l'operazione di rank, è possibile capire l'esone di provenienza di ciascun MEM.

Se i due MEM si trovano sullo stesso esone, il TIDMP è dato semplicemente dalla distanza tra la fine del primo MEM e l'inizio del secondo (sarà quindi uguale all' IDMP calcolato precedentemente).

Se i due MEM non si trovano sullo stesso esone, potrebbero essere su due esoni consecutivi o meno. Nel caso di esoni consecutivi, il TIDMP è dato dalla somma tra il suffisso non coperto dal primo esone e il prefisso non coperto dal secondo esone.

Il seguente frammento di codice mostra come calcolare il TIDMP:

```
def getTranscriptIdmp(transcripts, mems1, mems2, bv, exPos):
    tIdmp = 0

    # get last mem from read1 and first mem from read2
```

```

m1 = mems1[-1]
m2 = mems2[0]

# find exon for m1 and m2
id1 = bv.rank(m1[0]) - 1
id2 = bv.rank(m2[0]) - 1

# find positions on bitvector
start1 = m1[0]
start2 = m2[0]
len1 = m1[2]
len2 = m2[2]

if id1 == id2: # m1 and m2 are on same exon
    distance = start2 - (start1 + len1)
    tldmp += distance

else: # m1 and m2 are not on same exon
    consecutiveExons = False

# check in all transcripts if the exons are consecutive
# NOTE: two exons are consecutive if they appear
# next to each other in a transcript
# (an exon is represented as (startPosReference, endPosReference))
for _, exons in transcripts.items():
    start1Reference = getStart(m1, bv, exPos)
    start2Reference = getStart(m2, bv, exPos)
    end1Reference = getEnd(m1, bv, exPos)
    end2Reference = getEnd(m2, bv, exPos)
    for (s1, e1), (s2, e2) in pairwise(exons):
        if s1 == start1Reference and e1 == end1Reference
           and s2 == start2Reference and e2 == end2Reference:
            consecutiveExons = True
            exon1EndingPosBv = bv.select(id1+1)
            exon2startingPosBv = bv.select(id2)
            distance = exon1EndingPosBv - (start1 + len1)
                       + (start2 - exon2startingPosBv) - 1
            tldmp += distance
            break

if not consecutiveExons:
    pass

return tldmp

```

Il caso di esoni non consecutivi presenta diverse criticità, e per il momento non è stato calcolato.

5 Sviluppi futuri

5.1 Incremento della qualità degli allineamenti usando IDMP e TIDMP

Innanzitutto, è fondamentale ricordare che la distanza tra le due read di un pair viene fornita dall'ente che ha effettuato il sequenziamento, ed è la stessa per tutta la libreria. Qualora non fosse disponibile, è possibile calcolarla effettuando un allineamento con allineatori come BWA [8] o STAR [9], e lanciando un'analisi sul BAM ottenuto con SAM-Tools (ovviamente lo stesso risultato si otterrebbe usando ASGAL come allineatore, ma non avrebbe senso usarlo come parametro).

Questo valore potrebbe essere usato come riferimento per effettuare gli allineamenti in ASGAL. Quando un allineamento ha un IDMP (calcolato da ASGAL) troppo diverso da quello fornito in input, l'allineamento potrebbe essere scartato. Un'altra possibilità potrebbe essere quella di provare a ri-allineare cercando un IDMP più basso.

Un'altra possibilità potrebbe essere quella di utilizzare l'IDMP in fase di rilevazione, e attribuire un peso maggiore agli introni indotti da allineamenti (e quindi MEM) con un IDMP più simile a quello dato in input. Resta da formalizzare in che modo attribuire i pesi; questo approccio potrebbe inoltre avere conseguenze negative nella rilevazione di certi tipi di eventi.

Questi approcci hanno due problemi:

- Non sempre la distanza viene fornita
- Non tutti gli allineatori hanno la stessa nozione di "distanza tra read" (ad esempio alcuni considerano anche tutta la prima read e tutta la seconda nel conteggio) e, visto che utilizzano politiche di allineamento differenti, i valori ottenuti possono differire.

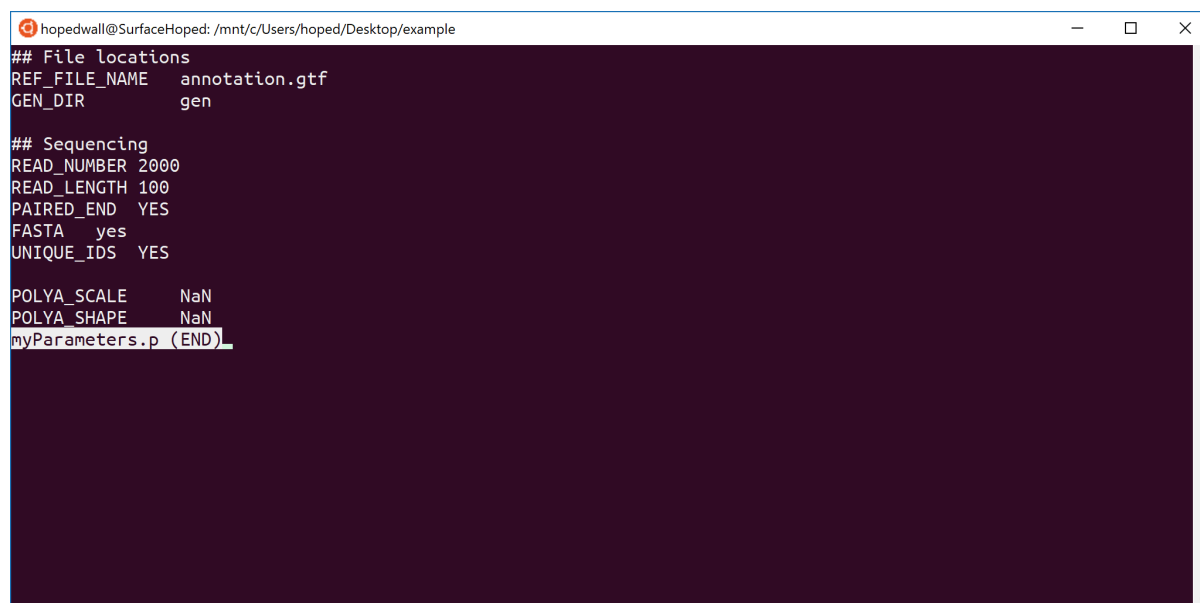
Per provare a risolvere questi problemi, abbiamo provato ad introdurre il TIDMP, una statistica calcolata ad-hoc da ASGAL. Come detto sopra, per il momento viene calcolata solo nel caso di esoni consecutivi, andando quindi a perdere informazioni negli altri casi. Inoltre, non siamo ancora sicuri dell'effettiva utilità della stessa. Questo aspetto dovrà essere approfondito in futuro.

6 Esempio di funzionamento

In questo esempio di funzionamento si utilizzerà il gene ENSG00000280145, situato sul cromosoma 21 dell' uomo (Homo Sapiens, release GRCh38/hg38). E' stato prima scaricato il genoma di riferimento da ensembl⁵ in formato fasta e la relativa annotazione in formato gtf. Dal file gtf (contenente l'annotazione per l'intero genoma) è stata isolata l'annotazione relativa al gene ENSG00000280145.

6.1 Generazione delle read con Flux Simulator

Si è scelto di utilizzare Flux Simulator [10] per la generazione delle read paired-end. Il suo utilizzo non è particolarmente complicato, ma è necessario passare i diversi parametri attraverso un file con estensione .p. Il file utilizzato in questa simulazione è il seguente:



```
hopedwall@SurfaceHoped: /mnt/c/Users/hoped/Desktop/example
## File locations
REF_FILE_NAME  annotation.gtf
GEN_DIR        gen

## Sequencing
READ_NUMBER    2000
READ_LENGTH    100
PAIRED_END     YES
FASTA          yes
UNIQUE_IDS     YES

POLYA_SCALE     NaN
POLYA_SHAPE     NaN
myParameters.p (END)
```

Figure 13: Il file contenente i parametri di Flux Simulator

Vengono così generati due file contenenti 2000 read di lunghezza 100, in formato fasta, che saranno dati in input ad ASGAL.

⁵<https://www.ensembl.org/index.html>

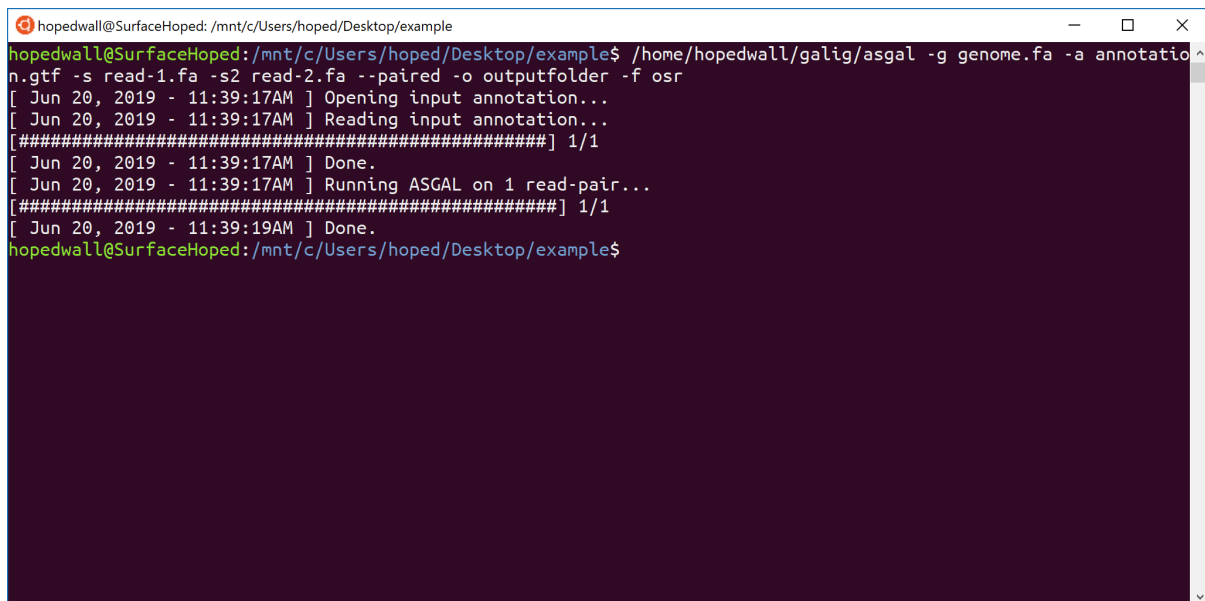
6.2 Utilizzo di ASGAL

ASGAL viene eseguito via linea di comando, richiamando lo script principale usando come parametri:

- Il **genoma** di riferimento (opzione **-g**)
- L'**annotazione** del genoma (opzione **-a**)
- I due file contenuti **read** (opzioni **-s** e **-s2**)
- La **cartella di destinazione** dell'output (opzione **-o**)
- L'indicazione delle **read paired-end** (opzione **-paired**)
- La **fragment library type** (opzione **-f**), opzionale per velocizzare la fase di allineamento

Questo script richiama nell'ordine lo Splice-Aware Aligner, il Formattatore SAM e il Rilevatore di eventi di Alternative Splicing, visualizzando alcune informazioni sul funzionamento.

Questa immagine mostra il funzionamento di ASGAL:



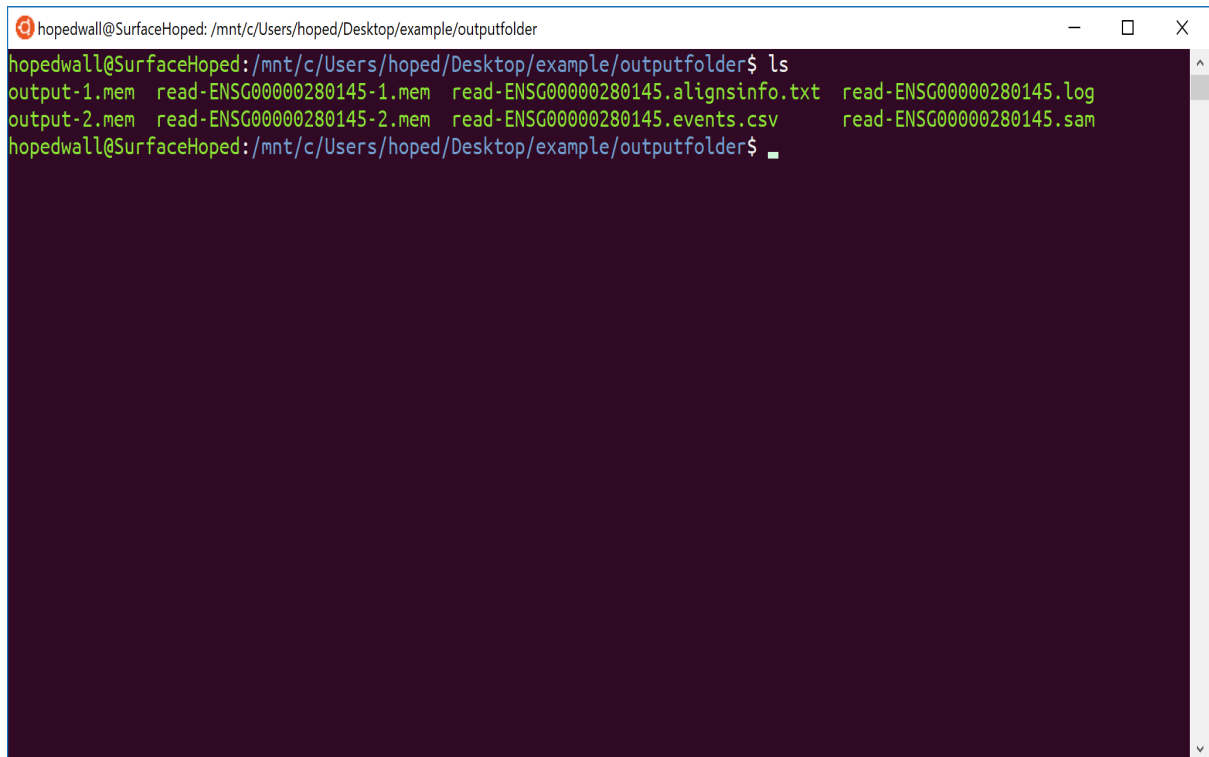
```
hopedwall@SurfaceHoped: /mnt/c/Users/hoped/Desktop/example
hopedwall@SurfaceHoped:/mnt/c/Users/hoped/Desktop/example$ /home/hopedwall/galig/asgal -g genome.fa -a annotation
n.gtf -s read-1.fa -s2 read-2.fa --paired -o outputfolder -f osr
[ Jun 20, 2019 - 11:39:17AM ] Opening input annotation...
[ Jun 20, 2019 - 11:39:17AM ] Reading input annotation...
[#####] 1/1
[ Jun 20, 2019 - 11:39:17AM ] Done.
[ Jun 20, 2019 - 11:39:17AM ] Running ASGAL on 1 read-pair...
[#####] 1/1
[ Jun 20, 2019 - 11:39:19AM ] Done.
hopedwall@SurfaceHoped:/mnt/c/Users/hoped/Desktop/example$
```

Figure 14: ASGAL in funzione

Sebbene sia possibile eseguire ciascuno script singolarmente, si raccomanda di usare lo script principale per un utilizzo più immediato.

6.3 Risultati

Dall'esecuzione vengono prodotti i seguenti file:



```
hopedwall@SurfaceHoped: /mnt/c/Users/hoped/Desktop/example/outputfolder
hopedwall@SurfaceHoped:/mnt/c/Users/hoped/Desktop/example/outputfolder$ ls
output-1.mem  read-ENSG00000280145-1.mem  read-ENSG00000280145.alignsinfo.txt  read-ENSG00000280145.log
output-2.mem  read-ENSG00000280145-2.mem  read-ENSG00000280145.events.csv      read-ENSG00000280145.sam
hopedwall@SurfaceHoped:/mnt/c/Users/hoped/Desktop/example/outputfolder$
```

Figure 15: I diversi file prodotti di ASGAL

Ovvero:

- I file .mem prodotti dal primo step
- Il file .sam e .alignsinfo.txt prodotti dal secondo step
- Il file .events.csv prodotti prodotto dal terzo step

Analizzando il file alignsinfo.txt si nota che oltre il 98% delle read sono state mappate.

```
hopedwall@SurfaceHoped: /mnt/c/Users/hoped/Desktop/example/outputfolder
Count mapped1: 1281/1313
Count mapped2: 1291/1295
Count unmapped1: 32
Count unmapped2: 4
Count placeholders1: 69
Count placeholders2: 87
Count mapped pairs: 1194
Count primary alignments: 958
Count secondary alignments: 236
Count one-side alignments: 184
read-ENSG00000280145.alignsinfo.txt (END)
```

Figure 16: Il file *alignsinfo.txt* contenente le informazioni sull'allineamento

Il file *events.csv* contiene gli eventi di Alternative Splicing rilevati da ASGAL:

```
hopedwall@SurfaceHoped: /mnt/c/Users/hoped/Desktop/example/outputfolder
Type,Start,End,Support,Transcripts
ES,6634769,6670521,7,ENST00000625185/ENST00000624965
A3,6630571,6634603,22,ENST00000623047
A3,6668244,6670521,11,ENST00000623324
A3,6668255,6670521,32,ENST00000623313
read-ENSG00000280145.events.csv (END)
```

Figure 17: Il file *events.csv* contenente gli eventi di Alternative Splicing rilevati

Sono stati rilevati tre eventi di Alternative Donor Site e un evento di Exon Skipping.

7 Competenze acquisite

Durante lo svolgimento dello stage sono state acquisite le seguenti competenze:

- Utilizzo di dati biologici in formati diversi (fasta, gtf, SAM, ecc.) e creazione di algoritmi che li manipolano
- Utilizzo di strumenti di natura bioinformatica: è stato utilizzato SAMTools per la validazione dei file SAM, IGV per la visualizzazione degli allineamenti sul genoma, gli allineatori BWA, STAR e BMap come riferimento per lo Splice-Aware Aligner, RNASeqSim e Flux Simulator per la generazione di read
- Utilizzo di community di esperti di bioinformatica: è stato usato Biostars per chiarire alcuni dubbi inerenti al funzionamento di SAMTools, il thread si trova a questo indirizzo: <https://www.biostars.org/p/376192/>
- Approfondimento dei linguaggi di programmazione Python e C++ e utilizzo di librerie specifiche per la bioinformatica (come ad esempio kseq per il parsing di file fasta)
- Utilizzo avanzato di Linux e di strumenti di environment management specifici per la bioinformatica (Bioconda)
- Utilizzo di strumenti di version control: è stato utilizzato Github, la relativa repo si trova a questo indirizzo: <https://github.com/HopedWall/galg>
- Lavoro di gruppo: è stato utilizzato il software Slack per comunicare con i membri dell'Algolab

8 Conclusioni

In questo documento sono state analizzate le modifiche apportate ad ASGAL, un tool per l'identificazione di eventi di Alternative Splicing espressi in un campione di RNA-Seq a partire da un genoma di riferimento e dall'annotazione di un gene, per introdurre il supporto alle read paired-end.

Sono state introdotte modifiche in ciascuno dei componenti di ASGAL: l' Allineatore Splice-Aware è ora in grado di allineare read paired-end e di farlo velocemente utilizzando le fragment library types; il Formattatore SAM produce file SAM coerenti con lo standard per read paired-end e mostra diverse statistiche sull'allineamento; il Rilevatore di eventi di Alternative Splicing è ora in grado di rilevare più eventi rispetto alla versione single-end.

Rimangono da investigare i possibili utilizzi di IDMP e TIDMP, due statistiche calcolate da ASGAL che rappresentano rispettivamente la distanza tra due read sul genoma e sui trascritti, che potrebbero portare ad un ulteriore incremento della capacità di rilevazione degli eventi di Alternative Splicing.

References

- [1] L. Denti, R. Rizzi, S. Beretta, G. Della Vedova, M. Previtali, and P. Bonizzoni, “Asgal: aligning rna-seq data to a splicing graph to detect novel alternative splicing events,” *BMC bioinformatics*, vol. 19, no. 1, p. 444, 2018.
- [2] S. Beretta, P. Bonizzoni, L. Denti, M. Previtali, and R. Rizzi, “Mapping rna-seq data to a transcript graph via approximate pattern matching to a hypertext,” in *International Conference on Algorithms for Computational Biology*, pp. 49–61, Springer, 2017.
- [3] E. Ohlebusch, S. Gog, and A. Kügel, “Computing matching statistics and maximal exact matches on compressed full-text indexes,” in *International Symposium on String Processing and Information Retrieval*, pp. 347–358, Springer, 2010.
- [4] E. T. Wang, R. Sandberg, S. Luo, I. Khrebtkova, L. Zhang, C. Mayr, S. F. Kingsmore, G. P. Schroth, and C. B. Burge, “Alternative isoform regulation in human tissue transcriptomes,” *Nature*, vol. 456, no. 7221, p. 470, 2008.
- [5] J. Tazi, N. Bakkour, and S. Stamm, “Alternative splicing and disease,” *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, vol. 1792, no. 1, pp. 14–26, 2009.
- [6] E. M. Rockenstein, L. McConlogue, H. Tan, M. Power, E. Masliah, and L. Mucke, “Levels and alternative splicing of amyloid β protein precursor (app) transcripts in brains of app transgenic mice and humans with alzheimer’s disease,” *Journal of Biological Chemistry*, vol. 270, no. 47, pp. 28257–28267, 1995.
- [7] H. Thorvaldsdóttir, J. T. Robinson, and J. P. Mesirov, “Integrative genomics viewer (igv): high-performance genomics data visualization and exploration,” *Briefings in bioinformatics*, vol. 14, no. 2, pp. 178–192, 2013.
- [8] H. Li, “Aligning sequence reads, clone sequences and assembly contigs with bwa-mem,” *arXiv preprint arXiv:1303.3997*, 2013.
- [9] A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras, “Star: ultrafast universal rna-seq aligner,” *Bioinformatics*, vol. 29, no. 1, pp. 15–21, 2013.
- [10] T. Griebel, B. Zacher, P. Ribeca, E. Raineri, V. Lacroix, R. Guigó, and M. Sammeth, “Modelling and simulating generic rna-seq experiments with the flux simulator,” *Nucleic acids research*, vol. 40, no. 20, pp. 10073–10083, 2012.