Algoritmi per la trasformata di Burrows–Wheeler posizionale con compressione run-length

Davide Cozzi
matr. 829827

Slide 1

Buongiorno, sono Davide Cozzi e oggi presento la mia tesi dal titolo: "Algoritmi per la trasformata di Burrows–Wheeler posizionale con compressione run-length"

Slide 2

Questa presentazione sarà così divisa:

- si vedrà una prima introduzione alla tematica della PBWT e del run-length encoding, al fine di poter comprendere al meglio gli scopi di questa tesi
- si vedranno poi i contributi in termini di strutture dati e algoritmi
- si vedranno poi i risultati sperimentali sui dati del 1000 Genome Project
- infine si chiuderà questa presentazione con qualche conclusione e con un breve excursus sugli sviluppi futuri di questo progetto

Slide 3.1

Negli ultimi anni si è assistito a un cambio di paradigma nel campo della bioinformatica, ovvero il passaggio dallo studio della sequenza lineare di un singolo genoma a quello di un insieme di genomi, provenienti da un gran numero di individui, al fine di poter considerare anche le varianti geniche. Questo nuovo concetto è stato introdotto da Tettelin, nel 2005, con il termine di pangenoma. Grazie ai risultati ottenuti in pangenomica, ci sono stati miglioramenti sia nel campo della biologia che in quello della medicina personalizzata, grazie al fatto che, con il pangenoma, si migliora la precisione della rappresentazione di multipli genomi e delle loro differenze. Il genoma umano di riferimento (GRCh38.p14), è composto da circa 3.1 miliardi di basi, con più di 88 milioni varianti tra i genomi sequenziati, secondo i risultati ottenuti nel 1000 Genome Project. Considerando che, grazie al miglioramento delle tecnologie di sequenziamento, la quantità dei dati di sequenziamento sia destinata ad aumentare esponenzialmente nei prossimi anni, risulta necessaria la costruzione di algoritmi e strutture dati efficienti per gestire una tale mole

di dati. Si hanno varie rappresentazioni possibili del pangenoma in termini di strutture dati. In primis abbiamo la concatenazione diretta di multipli genomi. Una soluzione più rappresentativa ed efficacie è tramite grafo, dove si da un ovvio peso alle variazioni tra i vari genomi (che condividono circa il 99% del codice genetico). Uno degli approcci più usati per rappresentare il pangenoma è attraverso un pannello di aplotipi, ovvero, da un punto di vista computazionale, una matrice di M righe, corrispondenti agli individui, e N colonne, corrispondenti ai siti con le varianti. Si specifica che, con il termine aplotipo, si intende l'insieme di alleli, ovvero di varianti che, a meno di mutazioni, un organismo eredita da ogni genitore. In questo contesto trova spazio uno dei problemi fondamentali della bioinformatica, ovvero quello del pattern matching, che sui pannelli di aplotipi sarà in questa tesi il calcolo dei set-maximal exact match. Inizialmente tale problema era relativo alla ricerca di una stringa (pattern) all'interno di un testo di grandi dimensioni, cioè il genoma di riferimento. Ora, con l'introduzione del pangenoma, il problema deve essere risolto sulle nuove strutture di rappresentazione del pangenoma.

Slide 3.2

Possiamo qui vedere un pannello di aplotipi con 20 sample/aplotipi e 15 siti con le varianti. In basso possiamo notare l'aplotipo query, ovvero il pattern, che presenta esattamente il numero di siti del pannello, avendo che ogni sito è esattamente lo stesso sito sul pattern. Da questo deriva il termine "posizionale" della PBWT.
In blu possiamo notare cosa si intende con SMEM, ovvero match massimali esatti tra una o più sottosequenze di aplotipi del pannello e una sottosequenza del pattern.
Si noti che si trattano pannelli costruiti su un alfabeto binario, a causa del fatto che la natura diploide dell'uomo comporta siti biallelici, anche se diversi studi segnalano una presenza stimata di 3/6% di siti triallelici (per uno SNP o anche per una delezione di una base).

Slide 4.1

Una delle strutture dati più utilizzata per la rappresentazione del pangenoma è la trasformata di Burrows–Wheeler Posizionale (PBWT), presentata da Richard Durbin nel 2014. Il funzionamento della PBWT prevede la costruzione di due insiemi di array, tramite l'ordinamento dei prefissi inversi a ogni colonna del pannello, detti insieme dei prefix array (che tiene traccia degli indici degli ordinamenti) e insieme dei divergence array (che tiene traccia della colonna d'inizio del prefisso inverso più lungo tra una riga e la precedente nel riordinamento ad una certa colonna). Il pannello, permutato tramite l'insieme degli a_k (che considerano l'ordinamento inverso fino alla colonna $k-1$ e), è detto matrice PBWT.
La PBWT, a causa della natura biologica del dato, avendo che il pannello è formato da aplotipi che tendono ad essere molto simili, tende a produrre nella matrice PBWT lunghe sequenze continue del medesimo carattere. Questo in quanto aplotipi simili fino alla colonna $k-1$ è molto probabile proseguano in colonna k col medesimo carattere.
Risulta quindi interessante il paradigma del run-length encoding, che consiste nel mem-

orizzare le run, ovvero sequenze massimali di caratteri uguali, in modo compatto, come coppia (carattere, lunghezza). Il punto fondamentale sarà rendere gli algoritmi che ora sono lineari sul numero di sample, ad essere sublineari sullo stesso, essendo lineari sul numero di run.
Possiamo qui vedere un semplice esempio con 000000 memorizzato come (0,6).

Slide 4.2

Qui, ad esempio, vediamo la costruzione della trasformata alla colonna 6, basata sul riordinamento fino alla quinta, e la conseguente produzione dei due array. Il prefix array tiene traccia degli indici riordinati, come visibile nella prima colonna dedicata agli indici mentre il divergence array della colonna d'inizio del prefisso inverso comune più lungo tra due sottosequenze consecutive nel riordinamento. Si noti che il divergence può anche essere sostituito dal Reverse Longest Common prefix, che memorizza la lunghezza del prefisso comune.

Slide 5

Il problema del calcolo degli SMEM presenta una soluzione semplice in $\mathcal{O}(N^2M)$, avendo che normalmente $N >> M$. Un tempo di calcolo assolutamente non permettente alcun tipo di analisi.
La PBWT permette di calcolare gli SMEM, di cui un esempio è qui disponibile, tra un aplotipo esterno e il pannello in tempo Avg. $\mathcal{O}(N + c)$ (dove c è il numero complessivo di SMEM), tramite il famoso algoritmo 5 di Durbin che si basa sul mantenere ed eventualmente estendere un intervallo sui prefix array che contiene gli indici delle righe che hanno uno SMEM fino a quella colonna. Se l'intervallo non è più estendibile si reporta lo SMEM e si sfrutta il divergence array per computare il nuovo intervallo. Il tradeoff di questo algoritmo è la richiesta in termini di spazio (13NM bytes), dovuto ad ulteriori array necessari in memoria per il "mapping", ovvero il forward step, tra una colonna e la successiva nella matrice PBWT (non computabili di volta in volta considerando di poter avere più query). Superare questo limite è l'obiettivo principale di questo progetto di tesi, presentando una PBWT run-length encoding in grado di computare gli SMEM in modo efficiente, dal punto di vista dello spazio. L'uso del run-length encoding potrebbe permettere una forte riduzione di memoria.

Slide 6

Si ha qui una breve panoramica delle componenti atomiche che hanno permesso la creazione delle varianti della RLPBWT:

- componenti per il mapping tra una colonna e la successiva nella PBWT, tramite bitvector sparsi (`MAP-BV`) o intvector compressi (`MAP-INT`). Tali componenti includono tutte le informazioni necessarie al mapping tra una colonna e la successiva, memorizzando l'indice di ogni run e gli equivalenti dell'FM-

index. Inoltre si memorizza un singolo bool per poter risalire la carattere di ogni singola run

- componenti per le threshold (`THR-BV/THR-INT`), dove si memorizza l'indice ogni threshold, ovvero il primo valore minimo dell'array RLCP
- componente per i prefix array sample (`PERM`), ovvero i valori di prefix array ad inizio e fine di ogni run
- componenti per il random access, tramite pannello di bitvector (`RA-BV`) tramite SLP (`RA-SLP`)
- componente per le LCE query con SLP (`LCE`)
- componente per il calcolo delle funzioni φ e φ^{-1} (`PHI`), che permettono data una colonna e un valore di prefix array, di sapere quale sia il valore precedente e quello successivo
- componente per il reverse longest common prefix (`RLCP`), che non scala sul numero di run

Il senso di queste multiple componenti si ritrova nel fatto che, parlando di strutture dati succinte e compresse, è difficile stimare l'effettivo spazio necessario basandosi solo sulle complessità asintotiche. Inoltre, dal punto di vista temporale, si aggiunge il problema che gli algoritmi tratti dipendono fortemente dalla caratteristica del dato. Al fine di esplorare a pieno le varie soluzioni quindi, oltre ad aver studiato e implementato le varianti di MONI e PHONI, si sono studiati gli usi di varie strutture dati sottostanti.
Si possono fare dei confronti tra le componenti con multiple rappresentazioni in termini di complessità temporale. Avendo che $M >> \rho$ si nota come gli intvector compressi si preannunciano più veloci. Si nota inoltre come il random access (o il calcolo delle LCE query), per quanto tale caso peggiore sia irrealistico nel nostro caso dovendosi in realtà basare sulla stringa prodotta dall'SLP, sia nettamente meno performante con tale grammatica compressa.

Slide 7

Possiamo qui confrontare velocemente a sinistra le stime di memoria dell'uso di un bitvector sparso e un intvector compressso, stime derivanti dai dati di SDSL (la lib usata) e dal numero di run attese (proporzionale a quanto visto con i dati del 1000 genome project). Si nota quindi che, per quanto all'inizio gli intvector compressi richiedano meno memoria tale comportamento è destinato ad invertirsi all'aumentare del numero di sample (assumendo la proporzionalità sperimentale tra M e ρ).

Slide 8.1

Per il calcolo degli SMEM si hanno due possibili soluzioni:

1. usare una variante dell'algoritmo di Durbin che però necessita del RLCP non proporzionale al numero di run, per le informazioni memorizzate. Inoltre non si è trovato un modo per calcolare quali righe presentassero uno SMEM ma solo quante.

Un'altra soluzione è usare l'array delle matching statistics

Definizione 1 *Dato un pannello X, di dimensioni $M \times N$, con M individui e N siti, e un aplotipo esterno/pattern z, tale che $|z| = N$, si definisce matching statistics di z su X un array* MS *di coppie* $(\mathsf{row}, \mathsf{len})$*, di lunghezza N, tale che (avendo che x_i indica l'i-esima riga del pannello X):*

- $x_{\mathsf{MS}[i].\mathsf{row}}[i-\mathsf{MS}[i].\mathsf{len}+1, i] = z[i-\mathsf{MS}[i].\mathsf{len}+1, i]$*, ovvero si ha che l'aplotipo query ha un match, terminante in colonna i, con la riga* $\mathsf{MS}[i].\mathsf{row}$
- $z[i - \mathsf{MS}[i].\mathsf{len}, i]$ *non è un suffisso terminante in colonna i di un qualsiasi sottoinsieme di righe di X. In altri termini, il match non deve essere ulteriormente estendibile a sinistra*

Tale soluzione permette il completo calcolo degli SMEM.

Lemma 1 *Dato un pannello X, di dimensioni $M \times N$, con M individui e N siti, un aplotipo esterno/pattern z, tale che $|z| = N$, e il corrispondente array di matching statistics* MS *si ha che $z[i - l + 1, i]$ presenta uno* SMEM *di lunghezza l in con la riga* $\mathsf{MS}[i].\mathsf{row}$ *del pannello X sse:*

$$\mathsf{MS}[i].\mathsf{len} = l \wedge (i = N - 1 \vee \mathsf{MS}[i].\mathsf{len} \geq \mathsf{MS}[i + 1].\mathsf{len})$$

Slide 8.2

A sua volta l'array MS può essere calcolato in due modi, ispirati da precedenti risultati ottenuti per la RLBWT:

1. usare le threshold per computare le righe e il random access per computare le lunghezze, come in MONI in due passaggi
2. usare le LCE query per computare entrambi in un solo passaggio, come in PHONI

Slide 9

Si ha quindi lo schema che mostra le 8 strutture dati studiate. Di fatto le soluzioni sono 3, che diventano 8 per il discorso di dualità visto precedentemente. Si hanno quindi:

1. le strutture che vediamo in centro, basate sul rifacimento dell'algoritmo 5 di Durbin e sull'uso dell'RLCP. Non è in grado di sapere quali siano le righe che presentano un certo SMEM ma solo quante

2. le soluzioni ispirate a MONI

3. le soluzioni ispirate a PHONI

Le ultime due soluzioni computano esattamente quali righe presentano uno SMEM.
È superfluo notare come le prime due soluzioni non siano effettivamente utilizzabili ma sono state citate in quanto punto iniziale di questa tesi e dei primi studi sull'uso del run-length.

Slide 10

In questa slide possiamo visualizzare un semplice esempio di matching statistics con la PBWT, con i avlori per le righe e per le lunghezze. Ad esempio, con $k = 5$, abbiamo $\mathsf{row} = 13$ e $\mathsf{len} = 6$, infatti con la riga 6 si ha un suffisso comune lungo 6, terminante in $k = 5$, con la riga 13.

Slide 11

Possiamo qui vedere, ad alto livello, come funzioni la struttura per il calcolo delle funzioni φ che permettono, dato un valore di prefix array e una colonna, di computare il valore di prefix array precedente e quello successivo. Il computo si basa sull'uso dei prefix array sample ed eventualmente dell'ultimo prefix array considerando quando due linee consecutive si separano durante le permutazioni. Quando si separano sono sicuramente una fine di run e una testa di run e quindi si può sapere che fino a quella colonna sono consecutive. Si tiene traccia quindi tramite bitvector di dove una riga sia testa o coda di run e tramite intvector compresso dell'indice della riga sopra e sotto. L'ultimo prefix array serve per quelle righe che non si "spezzano" mai. Quindi si cerca in su e in giù a partire dallo SMEM di MS fino a che si ha il medesimo SMEM (avendo che sono tutti consecutivi nel riordinamento in una certa colonna)

Slide 12

La sperimentazione, orchestrata tramite `snakemake`, è stata effettuata su una macchina con processore Intel Xeon E5-2640 V4 (2, 40GHz), 756GB di RAM, 768GB di swap e sistema operativo Ubuntu 20.04.4 LTS.
Si sono confrontate l'implementazione in C++ della RLPBWT e l'implementazione in C ufficiale della PBWT.
Si segnala che la RLPBWT supporta lo studio multithread di stringhe ma è stato usato un singolo thread per questi test a fini di avere risultati più comparabili.
Vediamo qui le caratteristiche dei pannelli usati, relativi alla phase 3 del 1000 Genome Project. Si nota come il numero di run sia molto inferiore all'altezza del pannello, fattore che conferma l'utilità del run-length encoding.

Slide 13

Si iniziano a vedere i risultati sperimentali.
In primis possiamo studiare i tempi di costruzione delle varie strutture anche se bisogna specificare che:

- nel caso della RLPBWT, per ognuna delle strutture dati composte, questa fase prevede la costruzione e la serializzazione dell'intera struttura dati
- nel caso della PBWT, questa fase crea unicamente un file compresso ad hoc, contenente le strutture base delle PBWT. A partire da tale file, in fase di calcolo degli SMEM, verranno calcolati anche tutti gli altri indici necessari al calcolo degli stessi, a seconda dell'algoritmo usato

Si confermano quindi i risultati attesi:

- usare il RLCP richiede più memoria
- l'uso dei bitvector richiede più memoria ma anche più tempo di costruzione (dovendo costruire le strutture per rank e select)

Slide 14

I risultati in termini di spazio richiesto dalle singole componenti della RLPBWT possono anche essere qui confermati, avendo appunto rappresentato il peso in mega di ognuna di esse:

- l'uso degli intvector è estremamente vantaggioso
- l'uso dell'SLP richiede pochissima memoria rispetto alla corrispondente con bitvector

Slide 15

Si confrontano ora i tempi di calcolo degli SMEM su 100 query. Parlando di PBWT si ha che matchIndexed è l'algoritmo 5 mentre si segnala che Durbin ha proposto anche un algoritmo per il calcolo degli SMEM tra un pannello di aplotipi e uno di query. Questo algoritmo è basato sulla creazione "virtuale" di un unico pannello, su cui costruire la PBWT, e sul calcolo dei match interni al pannello stesso, tale algoritmo è detto matchDynamic. Pur non essendo lo scopo della tesi studiare pannelli di query è sembrato corretto considerare tale soluzione.
I risultati sono evidenti:

- matchDynamic risulta essere il migliore sia in spazio (avendo che computa dinamicamente tutti gli indici dovendo farlo una sola volta per tutte le query) che in tempo
- si confermano i limiti spaziali dell'algoritmo 5 di Durbin

- si conferma l'uso in tempo e spazio degli intvector sui bitvector
- si conferma quanto detto in merito al random access

Aggiunegre qualche numero

Slide 16

Interessante è stato però confrontare i tempi su una singola query, calcolando media e deviazione standard su 100 query.
Si noti una cosa interessante: mathcDynamic ha quasi le stesse prestazioni su 1 query che su 100. Questo, aggiungendosi al fatto che i risultati non sono ordinati per query ma per permutazione, rappresenta un limite di tale algoritmo, ad esempio in ottica di uso tramite portale web (come per BLAST ora). Se per ogni query si carica e scarica diventa inefficiente.
In merito a tutte le altre soluzioni si conferma quanto già detto.
Aggiunegre qualche numero

Slide 17

Concludendo questa presentazione si segnala la potenzialità dei risultati run length encoded, confermando quanto già visto con la BWT classica. Inoltre l'obiettivo della tesi è stato pienamente raggiunto, come confermato dalla sezione risultati.
Ovviamente c'è spazio per molti sviluppi futuri:

- ottimizzazioni per pannelli di query, per mettere un avvicinamento ai risultati visti con matchDynamic
- SMEM interni con RLPBWT, per completare il quadro degli algoritmi di Durbin
- RLPBWT con dati mancanti, una tematica in forte studio in bioinformatica in quanto, causa errori nelle macchine di sequenziamento, ci sono siti per cui non si ha informazione. Studi in tal senso potrebbero portare ad una maggior potenza di imputazione
- RLPBWT multiallelica, per il discorso dei siti triallelici umani e per estendere lo studio, in futuro, anche ad altre specie (le piante sono fortemente poliploidi)
- calcolo K-SMEM con RLPBWT, ovvero SMEM con esattamente k righe, al fine di sperimentare la risoluzione di nuovi task utili a studi statistici

Slide 18

Per ulteriori dettagli e per altre strutture dati compresse sviluppate in ottica PBWT si rimanda al preprint del paper del quale sono autore. Questo progetto, infatti, è stato

svolto in collaborazione con il laboratorio BIAS e con diversi ricercatori internazionali (University of Florida Boucher-Rossi, Dalhousie University Gagie-Kashgouli e Tokyo Medical and Dental University Köppl).
Grazie a tutti per l'attenzione