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

Davide Cozzi
matr. 829827

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 prodotti in tali processi è destinata ad aumentare esponenzialmente nei prossimi anni, risulta necessaria la costruzione di algoritmi e strutture dati efficienti per gestire una tale informazione. A questo scopo, 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. 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 dello stesso.

Lo scopo di questa tesi è progettare strutture dati e algoritmi efficienti per risolvere il problema del pattern matching, inteso come ricerca dei set-maximal exact match (SMEM) tra un aplotipo esterno e un pannello di aplotipi, in una delle strutture dati più utilizzata per la rappresentazione del pangenoma: la *trasformata di Burrows–Wheeler posizionale* (PBWT). Il progetto di tesi, svolto in collaborazione con Prof. Gagie (Dalhousie University) e Prof.ssa Boucher (University of Florida), ha permesso lo sviluppo di una variante **run-length encoded** della PBWT, detta RLPBWT, che risolve il problema del calcolo degli SMEM tra un pannello di aplotipi e un aplotipo esterno, riducendo la memoria impiegata rispetto alla soluzione attualmente allo stato dell'arte.

Stato dell'arte

Gli algoritmi più efficienti per risolvere il problema del pattern matching tra un pattern e un testo sono attualmente basati sulla *trasformata di Burrows–Wheeler* (BWT), a sua volta strettamente legata ad altre due strutture dati: il suffix array (SA) e il longest common prefix (LCP).

Invece, allo scopo di risolvere il problema del calcolo degli SMEM tra un pannello di aplotipi e un aplotipo esterno, Durbin, nel 2014, propose una struttura dati ispirata alla BWT, detta *trasformata di Burrows–Wheeler posizionale* (PBWT). Tale trasformata viene costruita a partire da un pannello di aplotipi, rappresentato, riferendosi al solo caso bi-allelico, tramite una matrice binaria. La PBWT permette di calcolare gli SMEM tra un aplotipo esterno e il pannello in tempo *Avg.* $\mathcal{O}(N + c)$ (dove c è il numero complessivo di SMEM), mentre una soluzione semplice impiegherebbe $\mathcal{O}(N^2M)$. La motivazione essenziale della PBWT è considerare match, e quindi SMEM, dove anche le posizioni di inizio e fine sono rispettate. Tale vincolo, da cui deriva il termine "posizionale", non è soddisfacibile dalla BWT ed è dovuto al fatto che ogni colonna (o indice dell'aplotipo esterno) rappresenta un preciso sito per una specifica variante genica. Il tradeoff di questo algoritmo è la richiesta in termini di spazio: $13NM$ byte. Superare questo limite è l'obiettivo principale di questo progetto di tesi. 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* e *insieme dei divergence array*. Il pannello, permutato tramite l'insieme dei prefix array, è detto matrice PBWT.

Un altro obiettivo raggiunto negli ultimi anni è stato quello di studiare la costruzione di un'unica BWT partendo da multipli genomi. A tal fine, si è sfruttata una caratteristica della BWT, ovvero la tendenza a produrre una sequenza formata da run, ovvero sequenze massimali di caratteri uguali consecutivi. La causa di tale fenomeno si ritrova nelle ripetizioni di determinate sottostringhe nel testo, ripetizioni che sono frequenti se si studiano diverse sequenze genomiche concatenate. In letteratura è stata quindi proposta la *trasformata di Burrows–Wheeler run-length encoded* (RLBWT), dove ogni run viene memorizzata in modo efficiente come coppia `(carattere, lunghezza della run)`. Ad esempio, la stringa `aaaaaa` sarebbe memorizzata come `(a,6)`. Recentemente, per questa struttura dati compressa, è stato proposto un nuovo tipo di indicizzazione: il cosiddetto *r-index*. Tale indice riduce lo spazio di memoria richiesto in quanto non è lineare sulla lunghezza del testo ma sul numero di run della corrispondente BWT. L'r-index include la RLBWT e un suffix array sample, ovvero i valori di SA all'inizio e alla fine di ogni run. Tali studi hanno portato alla produzione di due tool, *MONI* e *PHONI*, alle cui tecniche è fortemente ispirata questa tesi. Queste due soluzioni hanno entrambe l'obiettivo di calcolare i maximal exact match (MEM) tra un testo e un pattern, ovvero sottostringhe del pattern che sono anche sottostringhe del testo, avendo che tale match non può essere esteso, in entrambe le direzioni, senza introdurre un mismatch.

Il calcolo dei MEM è direttamente correlato alla costruzione dell'array delle *matching statistics* (MS). Tale array, lungo quanto il pattern e formato da coppie (posizione pos, lunghezza len), annota, per ogni posizione i del pattern, la lunghezza len di un match, anche non massimale e iniziante in posizione i sul pattern, con la sottostringa nel testo

avente indice iniziale pos. Estraendo da tali match i MEM, si possono ottenere tutte le altre occorrenze del medesimo MEM nel testo. In MONI, per il calcolo delle MS in due passaggi sul pattern, è stato usato il concetto di *threshold*, definito come il primo valore minimo dell'array LCP tra due run consecutive dello stesso carattere. In PHONI, tramite l'uso delle LCE query per il calcolo dell'array MS, si è ridotto il calcolo delle MS in un singolo passaggio sul pattern. Infatti, tramite tale query, dati due indici del testo i e j, si restituisce il più lungo prefisso comune tra i suffissi i-esimo e j-esimo del testo.

Contributo

Questo lavoro di tesi è stato incentrato sullo sviluppo di approcci run-length encoded per la PBWT, ispirandosi ai risultati già ottenuti per la RLBWT. Essendoci diverse soluzioni possibili, si è deciso di suddividere il lavoro in varie componenti che permettessero l'assemblaggio di strutture dati composte atte al calcolo degli SMEM.

Alcuni di questi approcci si basano sull'uso dei bitvector sparsi, una struttura dati succinta che permette di memorizzare e interrogare vettori binari efficientemente. Infatti, tali strutture permettono due operazioni, dette rank e select, in tempo costante. La prima conteggia il numero di simboli $\sigma = 1$ fino ad una certa posizione mentre la seconda restituisce l'indice sul bitvector dell'i-esimo simbolo $\sigma = 1$. Le strutture necessarie all'indicizzazione delle run e al cosiddetto mapping, ovvero il "seguire" una riga dalla sua posizione permutata in una certa colonna alla posizione permutata nella colonna successiva, sono state implementate, anche, tramite bitvector sparsi (componente `MAP-BV`). Tali bitvector presentano un numero di simboli $\sigma = 1$ proporzionale al numero delle run. È stata anche proposta una struttura per il mapping basata sull'uso di intvector compressi, di lunghezza proporzionale al numero di run (componente `MAP-INT`).

Al fine di avvicinarsi alle idee proposte in MONI e PHONI, è stato necessario uno studio teorico preliminare per ridefinire: matching statistics (e conseguente calcolo degli SMEM), threshold e LCE query. In particolare, grazie agli ultimi due concetti, si è potuto evitare di memorizzare l'insieme dei divergence array. Le matching statistics sono definite tramite un array, lungo quanto l'aplotipo esterno e formato da coppie (riga row, lunghezza len), che, per ogni posizione, tiene traccia del suffisso comune che non sia estendibile a sinistra, terminante in quella colonna/posizione e lungo len, tra la riga row del pannello e l'aplotipo query. Calcolato tale array, è possibile estrarre uno SMEM tra l'aplotipo esterno e la riga row, terminante in posizione/colonna k. Infine, è possibile estendere tale risultato, tramite una struttura dati a supporto (componente `PHI`), a tutte le altre righe del pannello che presentano il medesimo SMEM. Il primo metodo di calcolo di tale array è stato basato sull'utilizzo delle threshold. Ragionando sulla matrice PBWT, una threshold è definita come l'indice del primo massimo valore del divergence array all'interno di una run (comprendendo anche la testa, qualora esistente, della run successiva, essendo il suo valore del divergence array calcolato tramite la coda della run corrente). L'insieme delle threshold, per una certa colonna, è stato memorizzato come bitvector sparso (componente `THR-BV`) oppure, come per il mapping, tramite un intvector compresso (componente `THR-INT`). Inoltre, si è provveduto a tenere in memoria i sample di prefix array a inizio e fine di ogni run (componente `PERM`).

Grazie all'uso delle threshold, si è potuto sviluppare un algoritmo efficiente, dal punto di vista della memoria richiesta, per il calcolo delle matching statistics in due "sweep" sull'aplotipo esterno, calcolandone prima i valori row e poi, tramite random access al pannello, i valori len. Tale soluzione richiede in memoria l'intero pannello, per il quale si è scelto di usare uno Straight-Line Program (SLP), ovvero una grammatica context-free che genera una e una sola parola, molto compressa in memoria, sulla quale è possibile effettuare random access (in tempo, da un punto di vista prettamente teorico, $\mathcal{O}(\log(NM))$, avendo un pannello con M aplotipi e N siti). Ad esempio, un pannello $4.908 \times 6.196.151$ richiede, in forma non compressa, circa 28GB di memoria, mentre l'SLP richiede solamente 0,2GB (garantendo random access). Per riferimento, comprimere il pannello con una tecnica standard (GZip) richiede 0,5GB. Oltre all'uso dell'SLP (componente `RA-SLP`), si è anche proposto l'uso di un insieme di bitvector per la memorizzazione del pannello (componente `RA-BV`), richiedendo maggior spazio in memoria (3,6GB nell'esempio precedente) ma minor tempo d'interrogazione ma garantendo random access in tempo costante.

È possibile risparmiare ulteriore spazio eliminando l'uso delle threshold e usando, per il calcolo delle matching statistics, le LCE query (componente `LCE`). Esse sono ridefinite, per la RLPBWT, come il suffisso comune più lungo tra due righe del pannello, eventualmente fissando il termine a una colonna precisata, e il loro calcolo è permesso in modo efficiente dall'uso dell'SLP (in tempo $\mathcal{O}(\log(NM))$). Calcolando la lunghezza di tale suffisso comune, è possibile computare i valori len delle matching statistics contemporaneamente al calcolo dei valori row, ottenendo il calcolo completo di tale array (e degli SMEM) in un singolo "sweep" sull'aplotipo esterno. Inoltre, sono sufficienti la coppia (row,len) corrente e quella precedente dell'array delle matching statistics per calcolare gli SMEM, riducendo ulteriormente lo spazio richiesto.

Risultati sperimentali

L'obiettivo della fase sperimentale è stato quello di confrontarsi con l'algoritmo di Durbin per il calcolo degli SMEM sulla PBWT al fine (1) di verificare se le soluzioni per la RLPBWT riducessero la memoria necessaria al calcolo degli SMEM e (2) di quantificare l'aumento dei tempi di calcolo. Infatti, per quanto riguarda (1), la caratterizzazione asintotica teorica non è sufficientemente stringente da garantire una riduzione pratica della memoria, in quanto fortemente dipendente dalle caratteristiche dei dati impiegati. Per quanto riguarda (2), un aumento dei tempi è atteso poiché la RLPBWT, a differenza dell'algoritmo di Durbin, utilizza strutture dati succinte e/o compresse che comportano tempi di calcolo superiori per le operazioni di accesso e/o interrogazione. I test sono stati effettuati su 5 pannelli della phase 3 del 1000 Genome Project, uno dei più importanti progetti di catalogazione delle varianti geniche umane. Tali pannelli sono relativi ai cromosomi 22, 20, 18, 16 e 1 (in ordine crescente di siti). Tutti i pannelli presentano 5.008 sample. Per questa sintesi, si è scelto di riportare alcuni risultati relativi al pannello del cromosoma 1, uno dei più grandi avendo dimensioni $4.908 \times 6.196.151$, che è stato interrogato con 100 query (estratte dal pannello originale), al fine di ridurre la fluttuazione statistica dei risultati. I risultati qui riportati, relativi alla PBWT, sono stati ottenuti us-

ando l'implementazione ufficiale della stessa. Per la RLPBWT si riportano solo i risultati dell'implementazione con LCE query e mapping tramite intvector compressi, essendo la soluzione con minor richiesta di memoria.

In merito ai tempi di calcolo, si è ottenuto il risultato atteso: con la PBWT si ha il calcolo degli SMEM in 1.026s mentre con la RLPBWT in 1.142s, avendo una differenza, per quanto attesa, molto piccola. Si è passato, quindi, ad analizzare i risultati in termini di memoria. Secondo le stime di Durbin, gli array necessari al funzionamento del suo algoritmo richiederebbero $13NM$ byte, implicando, per il pannello in analisi, circa 368GB di memoria. Sperimentalmente, si sono registrati picchi di memoria prossimi ai 369GB, confermando le stime. Invece, per la variante della RLPBWT presa in esame, il picco registrato è stato di appena 4GB, ottenendo una riduzione di memoria di quasi il 99%. Tale riduzione renderebbe possibile l'esecuzione del calcolo anche su un comune portatile, assumendo di aver a disposizione l' SLP .

Conclusioni e sviluppi futuri

In conclusione, le strutture dati composte per la RLPBWT, basate sul calcolo dell'array delle matching statistics, sono state tutte in grado, al variare della soluzione, di ridurre sensibilmente la quantità di memoria necessaria a risolvere il problema del calcolo degli SMEM tra un pannello di aplotipi e un aplotipo esterno.

Seppur questa tesi abbia raggiunto l'obiettivo prefissato, il lavoro può essere ulteriormente esteso in diverse direzioni, ad esempio ottimizzando le strutture, sia in termini di gestione del mapping che, eventualmente, di gestione di più query contemporaneamente. In merito, 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 implementazione, nel setup di test descritto nella sezione precedente (che comporta il perfetto input per tale soluzione), risulta essere molto performante sia in termini di tempo (93s) che di memoria utilizzata (0,1GB) durante l'esecuzione. Si segnala che, per quanto riguarda i tempi di calcolo, tale algoritmo diventa inefficiente, rispetto alle altre soluzioni proposte, al diminuire del numero di query.

Inoltre, sono possibili diverse generalizzazioni rispetto alle caratteristiche del pannello, considerando, ad esempio, pannelli multiallelici o con dati mancanti (casi comuni in dati reali). In particolare, la gestione di dati mancati, molto rilevante a causa della non perfezione delle tecnologie di sequenziamento, risulta essere un problema aperto in bioinformatica. Quindi, sarà richiesto lo sviluppo di nuove metodologie, basate, ad esempio, su algoritmi parametrici o algoritmi approssimati, per la risoluzione del problema del calcolo degli SMEM su tali pannelli.

Le potenzialità di tale struttura dati sono molteplici e, grazie al ridotto consumo di memoria, si hanno le giuste premesse perché venga utilizzata per gestire e interrogare grandi moli di dati reali, incrementando le capacità di studio, previsione e inferenza che si possono avere per mezzo dello studio del pangenoma.