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 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. 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 del pangenoma.

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. Boucher (University of Florida), ha quindi permesso lo sviluppo di una variante **run-length encoded** della **PBWT**, detta **RLPBWT**, che risolve questo problema 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)*. Infatti, è stato dimostrato che la BWT è fortemente legata a due strutture dati fondamentali per il pattern matching: il Suffix Array (SA), cioè la lista delle posizioni di partenza di tutti i suffissi di un testo presi in ordine lessicografico, e il Longest Common Prefix (LCP), cioè la lista delle lunghezze del più lungo prefisso comune tra ogni coppia di suffissi consecutivi indicizzati nel SA.

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 $\mathcal{O}(MN)$, mentre una soluzione semplice impiegherebbe $\mathcal{O}(M^2N)$. La motivazione essenziale della PBWT è considerare match, e quindi anche 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 della query) rappresenta un preciso sito per una specifica variante genica. Il tradeoff di questo algoritmo è la richiesta in termini di spazio ($13NM$ bytes). 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*. Per ogni colonna, secondo l'ordinamento dettato dalla stessa, i due array presentano un comportamento analogo a quanto visto per il suffix array e l'array LCP nella BWT. Il pannello permutato tramite i valori dell'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 con caratteri uguali in posizioni consecutive, a causa delle ripetizioni di determinate sottostringhe nel testo. Partendo da molte sequenze genomiche concatenate si hanno, ovviamente, molte regioni ripetute che comportano tali sequenze massimali di caratteri uguali nella BWT, dette run. 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 perché non è lineare sulla lunghezza del testo ma sul numero di run della sua BWT. L'r-index include la RLBWT e un suffix array sample, ovvero i valori del SA all'inizio e alla fine di ogni run. Tali articoli 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.

Si noti che 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 del pattern, la lunghezza *len* di un match, anche non massimale, con una sottostringa nel testo avente indice iniziale *pos*. Estraendo da tali match i MEM si possono poi ottenere tutte le altre occorrenze del medesimo MEM nel testo. In MONI, per il calcolo delle MS, è stato usato anche il concetto di *threshold*, definito come il minimo valore dell'array LCP tra due run consecutive dello stesso carattere. Un ulteriore miglioramento si è registrato in *PHONI* con l'uso delle *LCE query* per il calcolo dell'array MS. Infatti, tale struttura, dati due indici del testo i e j, restituisce il più lungo prefisso comune tra il suffisso i-esimo e il suffisso j-esimo del testo.

Si noti, infine, un collegamento naturale che si ha tra la PBWT e la BWT, avendo, ad esempio, che il prefix array, in una certa colonna, della prima corrisponde al suffix array della seconda mentre il divergence array è una diversa rappresentazione dell'array LCP. Anche grazie a queste premesse, si è basato il progetto di tesi sulla costruzione RLPBWT a partire dalle tecniche teorizzate per la RLBWT.

Contributo

L'idea con la quale si è sviluppata questa tesi vede, come punto di partenza, alcuni primi risultati teorici, presenti in letteratura, che caratterizzavano una variante run-length encoded della PBWT. Tuttavia, tale proposta era inefficiente in termini di memoria e su di essa non era possibile il calcolo degli SMEM. Infatti, un adattamento dell'algoritmo per il calcolo degli SMEM proposto da Durbin all'uso con una struttura run-length encoded non sembra essere possibile mantenendo lo spazio impiegato per memorizzare i dati del divergence array proporzionale al numero di run. Inoltre non sembra possibile trovare una soluzione efficiente per utilizzare i sample del prefix array ad inizio e fine di ogni run.

Nel lavoro di tesi si è quindi deciso di cambiare approccio al problema, ispirandosi ai risultati già ottenuti per la RLBWT. Tale approccio si basa anche 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 permette di conteggiare il numero di simboli $\sigma = 1$ fino ad una certa posizione mentre la seconda di ottenere l'indice sul bitvector dell'i-esimo simbolo $\sigma = 1$. Quindi, 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 in quella successiva nella matrice PBWT, sono state implementate tramite bitvector sparsi, con un numero di simboli $\sigma = 1$ proporzionale al numero run. Al fine di avvicinarsi alle idee proposte in *MONI* e *PHONI* è quindi servito 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 il divergence array. Le matching statistics sono definite tramite un array, lungo quanto il pattern e formato da coppie (riga *row*, lunghezza *len*),

che, per ogni colonna, tiene traccia del suffisso comune che non sia estendibile a sinistra, terminate in quella colonna e lungo *len*, tra la riga *row* del pannello e l'aplotipo query. Una volta calcolato tale array è possibile estrarre uno SMEM, terminate in colonna k tra la query e la riga *row*, ed estendere, tramite una struttura dati a supporto, tale risultato 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 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 sfruttando anche la coda della run corrente). L'insieme delle threshold per una certa colonna è stato memorizzato come bitvector sparso. Inoltre, si è provveduto a tenere in memoria i sample di prefix array a inizio e fine di ogni run. 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" sul pattern, calcolandone prima le posizioni e poi, tramite random access al pannello, anche le lunghezze. Tale soluzione richiedeva 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 compatta in memoria, sulla quale è possibile effettuare random access (in tempo $\mathcal{O}(\log s)$ dove s è la lunghezza della parola prodotta tramite l'SLP). Ad esempio, un pannello 70.000×46.538 richiede per essere rappresentato in forma non compressa circa 3,3GB di memoria, mentre tramite SLP richiede solamente in 7MB. Per riferimento, comprimere il pannello con una tecnica standard (GZip) richiede 451MB.

Inoltre, si è mostrato che è possibile risparmiare ulteriore spazio eliminando l'uso delle *threshold*. Per ottenere il calcolo dell'array delle matching statistics si sono quindi usate le LCE query, ridefinite in questo caso come il suffisso comune più lungo tra due righe del pannello, eventualmente fissando il termine a una colonna precisata. Il loro calcolo è permesso in modo efficiente dall'uso dell'SLP. Calcolando la lunghezza di tale suffisso comune, è possibile computare anche le lunghezze delle matching statistics contemporaneamente al calcolo delle righe. Quindi, si è ottenuto il calcolo completo di tale array in un singolo "sweep" sul pattern, dimezzando il numero di passaggi sulla struttura dati. Si noti che il calcolo degli SMEM viene fatto contemporaneamente al calcolo delle lunghezze delle matching statistics quindi, con l'uso delle LCE query, si è ridotto sia lo spazio necessario in memoria per la RLPBWT che il tempo di calcolo degli SMEM.

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 la RLPBWT riducesse 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 in pratica, in quanto fortemente dipendente dalle caratteristiche dei dati impiegati. Per quanto riguarda (2), un aumento dei tempi è atteso in quanto la RLPBWT, a differenza dell'algoritmo di Durbin, utilizza

strutture dati succinte che comportano tempi di calcolo superiori per le operazioni di accesso e/o interrogazione. I test sono stati effettuati su pannelli simulati realisticamente tramite uno dei software più usati in bioinformatica per generare pannelli di aplotipi (MaCS). Per questa sintesi, si è scelto di riportare alcuni risultati relativi ad un pannello 70.000×46.538, che è stato interrogato con 30.000 query, al fine di ridurre la fluttuazione statistica dei risultati. I risultati qui riportati, relativi alla PBWT, sono stati ottenuti usando l'implementazione ufficiale della stessa. Per la RLPBWT, invece, si è utilizzata l'implementazione con le LCE query.

In merito al tempo di calcolo si è ottenuto il risultato atteso: con la PBWT si ottiene il calcolo degli SMEM in 411s mentre con la RLPBWT in 1824s, ovvero poco più del quadruplo del tempo. 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 $\sim 13NM$ bytes, implicando, per il pannello in analisi, circa 40GB di memoria. Anche sperimentalmente si sono registrati picchi di memoria prossimi ai 40GB di memoria, confermando le stime. Invece, per la RLPBWT, il picco registrato è stato di appena $\sim$ 3GB, cioè si è ottenuta una riduzione del 92,5% di memoria rispetto alla soluzione allo stato dell'arte.

Conclusioni e sviluppi futuri

In conclusione, la RLPBWT proposta è stata in grado di ridurre sensibilmente la quantità di memoria necessaria per la risoluzione del problema del calcolo degli SMEM tra un pannello di aplotipi e un aplotipo esterno.

Seppure questa tesi abbia raggiunto l'obiettivo prefissato, il lavoro può essere ulteriormente esteso in diverse direzioni. Tra di esse si ha un'ulteriore ottimizzazione della struttura, sia in termini di gestione del mapping che, eventualmente, di gestione di più query contemporaneamente. In merito a quest'ultimo sviluppo si segnala che Durbin ha proposto anche un algoritmo per il calcolo degli SMEM tra un pannello di aplotipi e un pannello di query. Questo algoritmo è basato sulla creazione 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 si noti comportare il perfetto input per tale soluzione), risulta essere molto performante sia in termini di tempo (22s) che di memoria utilizzata (10.084KB) durante l'esecuzione.

Inoltre, sono possibili diverse generalizzazioni rispetto alle caratteristiche del pannello, come ad esempio pannelli multi-allelici o con dati mancanti. Tali caratteristiche sono, infatti, 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. Sarà quindi richiesto lo sviluppo di nuove metodologie, basate, eventualmente, su algoritmi parametrici o algoritmi approssimati, per la risoluzione del problema del calcolo degli SMEM su pannelli di questo tipo.

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