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

Davide Cozzi

Relatore: *Prof.ssa Raffaella Rizzi* **Correlatore:** *Dr. Yuri Pirola*

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
Università degli Studi di Milano Bicocca

26 Ottobre 2022

Outline

Outline

Un punto di vista per il pangenoma

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**.

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.

Un **aplotipo** è l'insieme di alleli, ovvero di varianti che, a meno di mutazioni, un organismo eredita da ogni genitore.

Scopo della tesi

Lo scopo di questa tesi, svolta in collaborazione con il laboratorio BIAS e con diversi ricercatori internazionali (University of Florida, Dalhousie University e Tokyo Medical and Dental University), è stato quello di creare una variante **run-length encoded** della **trasformata di Burrows–Wheeler posizionale** che permettesse, in modo efficiente dal punto di vista della memoria richiesta, il calcolo dei **set-maximal exact matches** di un aplotipo esterno contro un pannello biallelico di aplotipi.

Ideare una struttura dati efficiente in spazio permetterà, nel breve futuro, di gestire i sempre più grandi dati provenienti dal sequenziamento comportando più precisi studi di imputazione coi GWAS etc...

Outline

Alcuni concetti teorici

- bitvector, bitvector sparsi e intvector compressi
- straight-line program (SLP) e longest common extension (LCE) query
- trasformata di Burrows–Wheeler (BWT), (inverse) suffix array ((I)SA), (permuted) longest common prefix ((P)LCP), FM-index, LF-mapping, funzione φ, maximal exact matches (MEM) e backward-search
- trasformata di Burrows–Wheeler run-length encoded (RLBWT), r-index, Toehold lemma e matching statistics (MS)
- BWT posizionale (PBWT) e set-maximal exact match (SMEM)

La BWT (PBWT) tendono ad avere caratteri uguali in posizioni consecutive all'interno della sua sequenza (ogni sua colonna). Si usa il run-length encoding che consiste nel memorizzare le cosiddette run, ovvero sequenze massimali di caratteri uguali, mediante coppie `(carattere, lunghezza della run)`, ad esempio la stringa `aaaaaa` sarebbe memorizzata come `(a,6)`.

MONI e PHONI

MONI

Rossi et al., nel 2021, sfruttarono le conoscenze relative alla RLBWT e all'r-index per ideare **MONI**. In questa soluzione si ha la costruzione, in due sweep, tramite l'uso delle threshold (*algoritmo di Bannai*), dell'array delle matching statistics, da cui si computano i maximal exact match.

Rossi et al.: *MONI: A pangenomic index for finding maximal exact matches, 2021*

PHONI

Nel 2021, Boucher, Gagie, Rossi et al. proposero un ulteriore miglioramento di quanto fatto in MONI, con **PHONI**, usando le LCE query al posto delle threshold, ottenendo un algoritmo "online".

Boucher et al.: *PHONI: Streamed matching statistics with multi-genome references, 2021*

PBWT

X	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
14	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
15	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
00	1	0	0	1	0	0	0	0	0	0	0	1	1	0	1
09	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
10	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
16	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1
08	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1
11	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0
12	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
13	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
18	0	1	1	0	1	0	0	0	0	0	0	1	0	0	1
19	0	1	1	0	1	0	1	0	0	0	0	0	1	0	1
01	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1
02	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
03	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
17	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1
04	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
05	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
06	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
07	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1

$a_6 = [14, 15, 0, 9, 10, 16, 8, 11, 12, 13, 18, 19, 1, 2, 3, 17, 4, 5, 6, 7]$

$d_6 = [6, 0, 4, 2, 0, 0, 5, 0, 0, 0, 3, 0, 4, 0, 0, 6, 4, 0, 0, 0]$

Durbin: *Efficient haplotype matching and storage using the positional Burrows–Wheeler transform (PBWT), 2014*

Set-maximal exact match

Calcolo SMEM via **algoritmo 5 di Durbin** in tempo $\mathcal{O}(NM) + \text{Avg.}\mathcal{O}(N + c)$, richiedendo 13NM byte

X	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
00	1	0	0	1	0	0	0	0	0	0	0	1	1	0	1
01	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1
02	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
03	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
04	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
05	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
06	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
07	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1
08	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1
09	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
10	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
11	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0
12	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
13	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
14	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
15	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
16	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1
17	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1
18	0	1	1	0	1	0	0	0	0	0	0	1	0	0	1
19	0	1	1	0	1	0	1	0	0	0	0	0	1	0	1
z	0	1	0	0	1	0	1	0	0	0	1	1	1	0	1

Outline

Le componenti

Componenti innovative derivate dallo studio della RLBWT in ottica PBWT

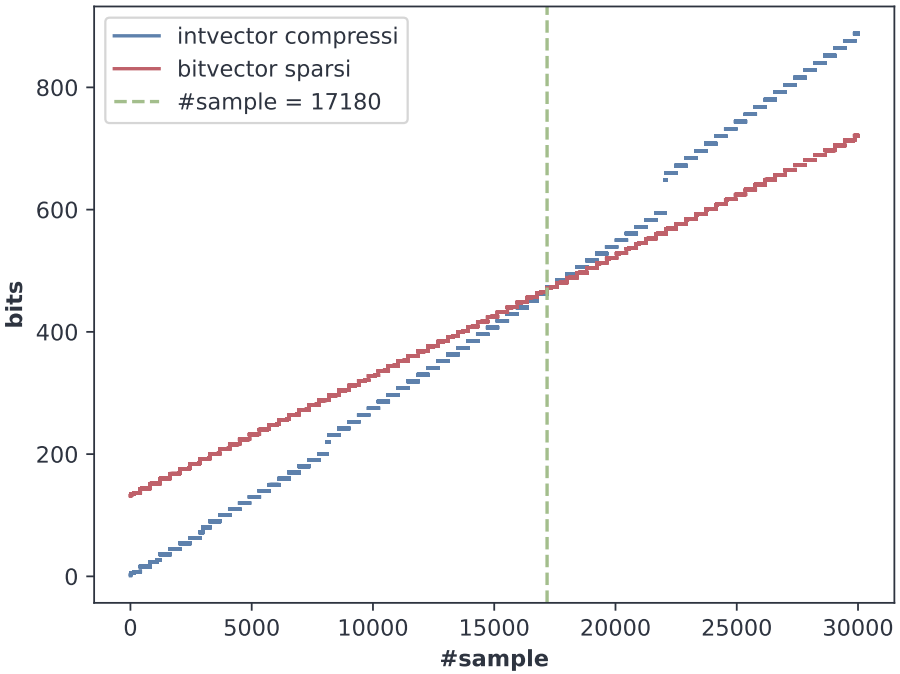
- componenti per il mapping tra una colonna e la successiva nella PBWT, tramite bitvector sparsi (`MAP-BV`) o intvector compressi (`MAP-INT`)
- componenti per le threshold (`THR-BV`/`THR-INT`)
- componente per i prefix array sample (`PERM`)
- componenti per il random access, tramite pannello di bitvector (`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`)
- componente per il reverse longest common prefix (`RLCP`)

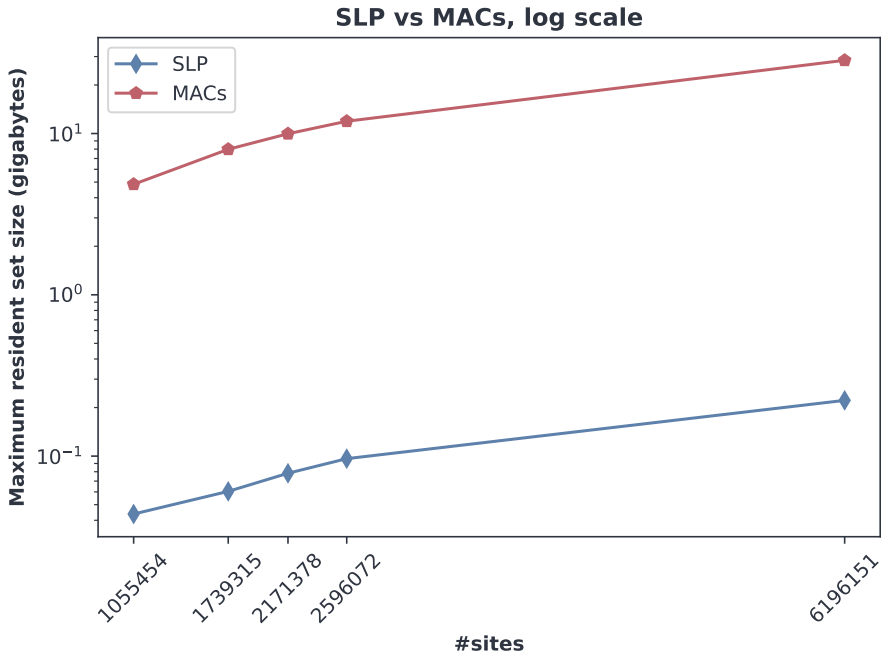
Alcune complessità

Complessità in tempo, pannelli $N \times M$, con ρ numero di run in una colonna

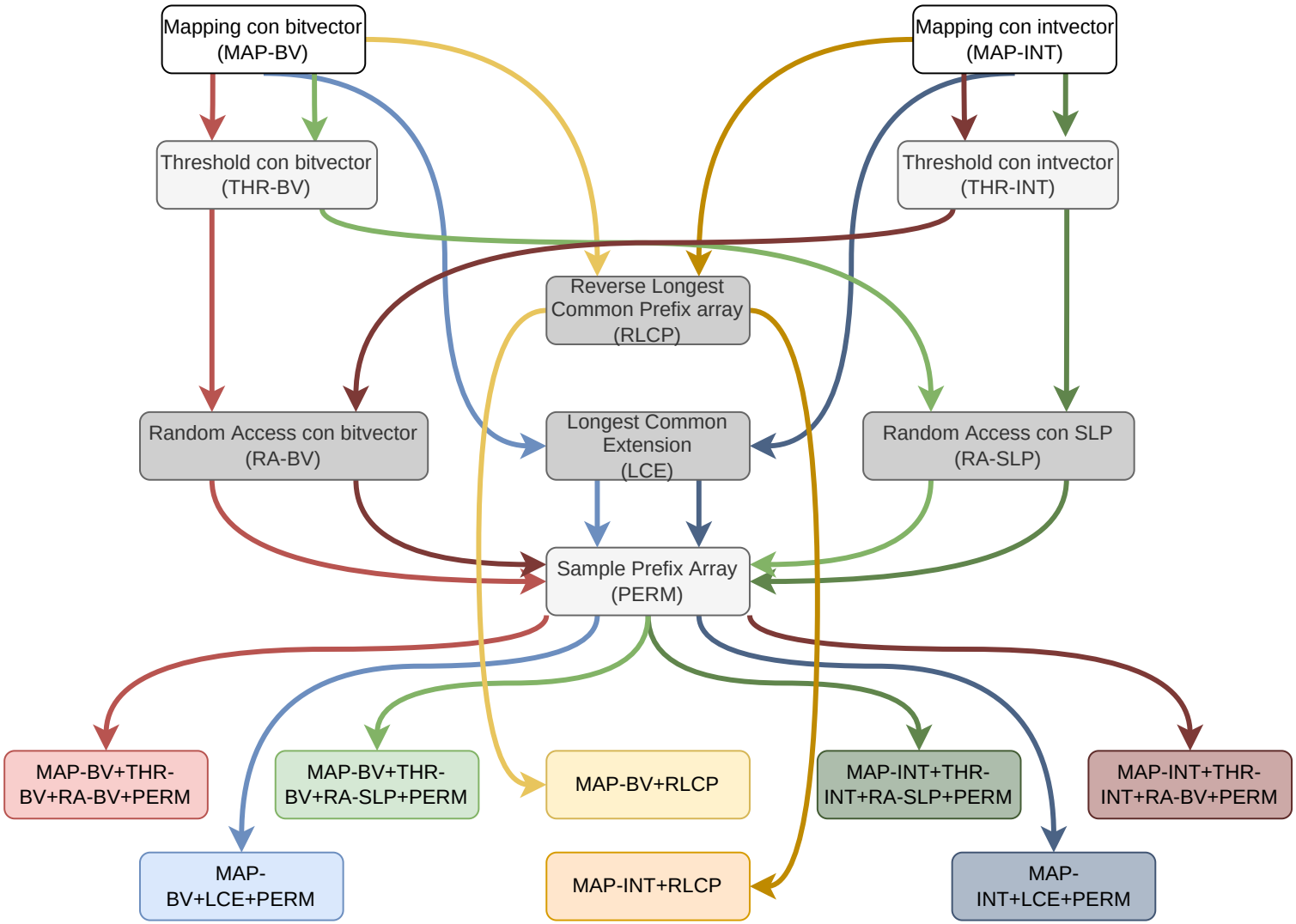
- funzione rank con bitvector sparsi: $\mathcal{O}\left(\log\left(\frac{M}{\rho}\right)\right)$
- funzione select con bitvector sparsi: $\mathcal{O}(1)$
- equivalente della funzione rank con intvector compressi: $\mathcal{O}(\log(\rho))$
- equivalente della funzione select con intvector compressi: $\mathcal{O}(1)$
- random access con `RA-BV`: $\mathcal{O}(1)$
- random access con `RA-SLP` e calcolo LCE query: $\mathcal{O}(\log(NM))$

Qualche confronto in spazio





Componenti e strutture dati, una panoramica



Calcolo degli SMEM

Due macro soluzioni

1. usare l' RLCP, adattando l'algoritmo 5 di Durbin. Tale soluzione non permette di sapere quali righe del pannello in input presentino uno SMEM fino ad una certa colonna ma solo quante
2. usare l'array MS. Tale soluzione permette di riconoscere ogni riga del pannello in input che presenti uno SMEM fino ad una certa colonna

Calcolo degli SMEM

Due macro soluzioni

1. usare l' RLCP, adattando l'algoritmo 5 di Durbin. Tale soluzione non permette di sapere quali righe del pannello in input presentino uno SMEM fino ad una certa colonna ma solo quante
2. usare l'array MS. Tale soluzione permette di riconoscere ogni riga del pannello in input che presenti uno SMEM fino ad una certa colonna

Due macro alternative per il calcolo dell'array MS

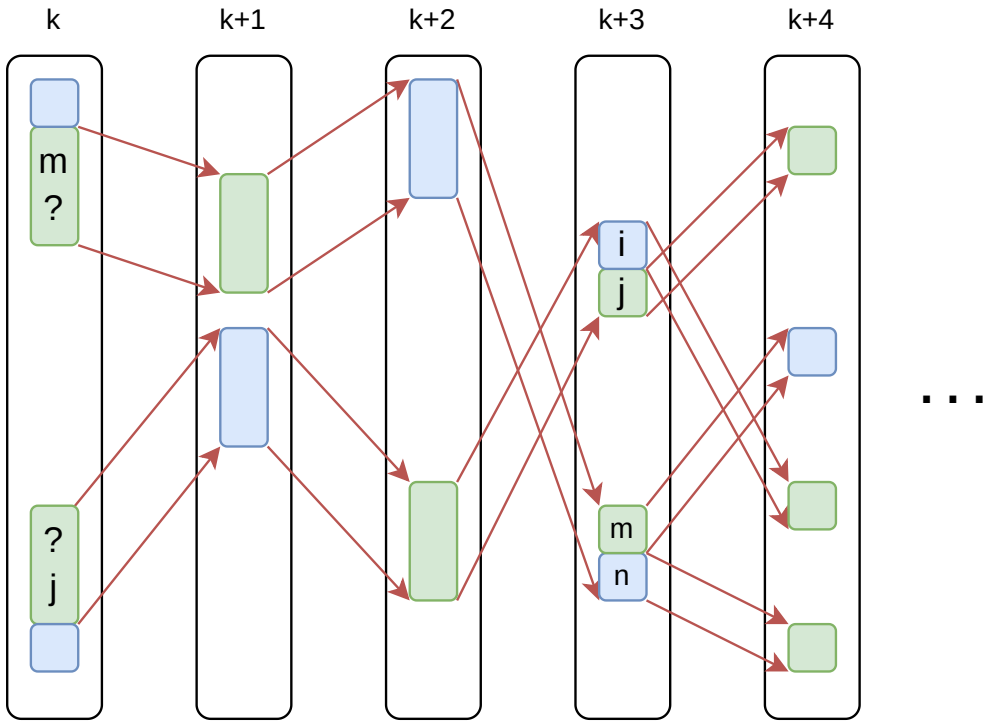
1. usare le threshold e il random access al pannello (come in MONI)
2. usare le LCE query (come in PHONI)

Matching statistics

X	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
00	1	0	0	1	0	0	0	0	0	0	0	1	1	0	1
01	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1
02	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
03	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
04	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
05	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
06	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
07	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1
08	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1
09	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
10	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
11	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0
12	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
13	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
14	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
15	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
16	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1
17	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1
18	0	1	1	0	1	0	0	0	0	0	0	1	0	0	1
19	0	1	1	0	1	0	1	0	0	0	0	0	1	0	1
z	0	1	0	0	1	0	1	0	0	0	1	1	1	0	1

k	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
row	19	19	16	15	13	13	19	19	19	19	11	11	17	17	17
len	1	2	3	4	5	6	4	5	6	7	4	5	2	3	4

Struttura per le funzioni φ e φ^{-1}



$$\Phi_j = [0,0,0,1,0,\ldots], \quad \Phi_m^{-1} = [0,0,0,1,0,\ldots], \quad \Phi_{supp} = [i,\ldots], \quad \Phi_{supp}^{-1} = [n,\ldots]$$

$$\Phi^j_{supp}[\text{rank}^{\varphi}_j(0)] = \Phi^j_{supp}[0] = i, \quad \Phi^{-1\ m}_{supp}[\text{rank}^{\varphi^{-1}}_m(0)] = \Phi^{-1\ m}_{supp}[0] = n$$

Outline

Sperimentazione e dati

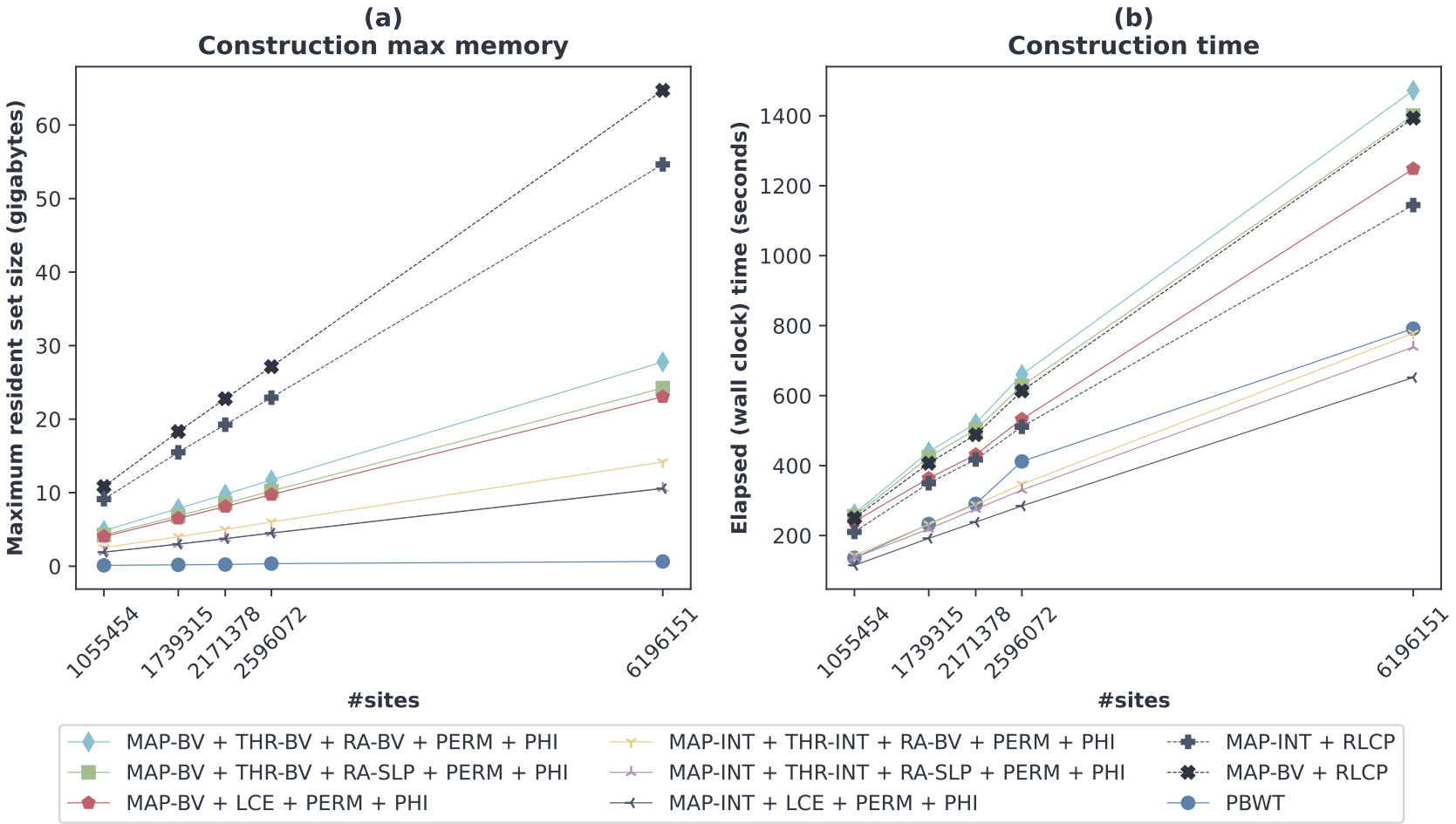
Implementazione e sperimentazione

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.

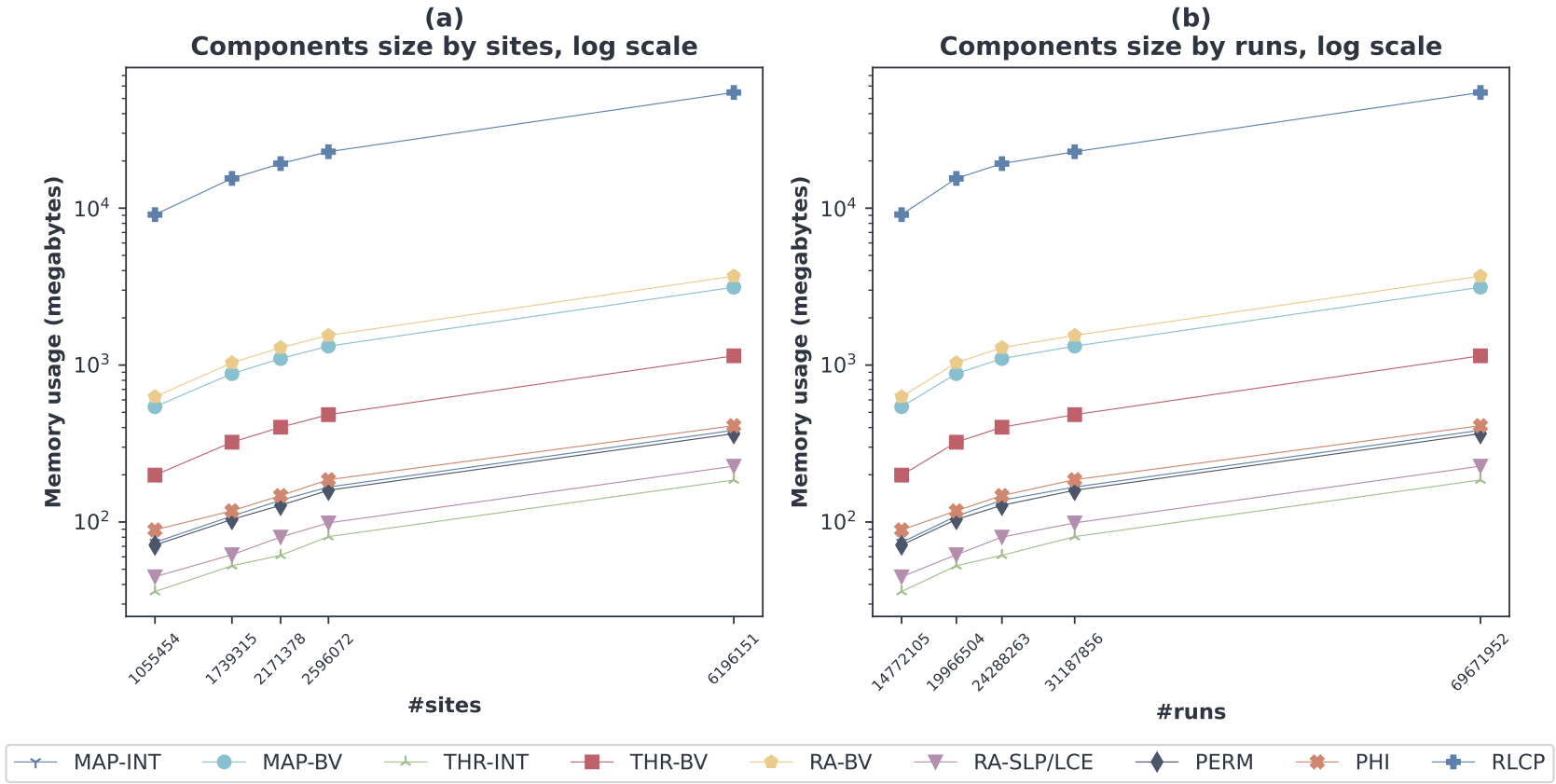
Pannelli del 1000 Genome Project con 4908 sample, avendone estratti 100 come query.

Chr	#Siti	#Run totale	Max run	Media run
chr22	1.055.454	14.772.105	2.450	14
chr20	1.739.315	19.966.504	2.176	11
chr18	2.171.378	24.288.263	2.365	11
chr16	2.596.072	31.187.856	2.330	12
chr1	6.196.151	69.671.952	2.721	11

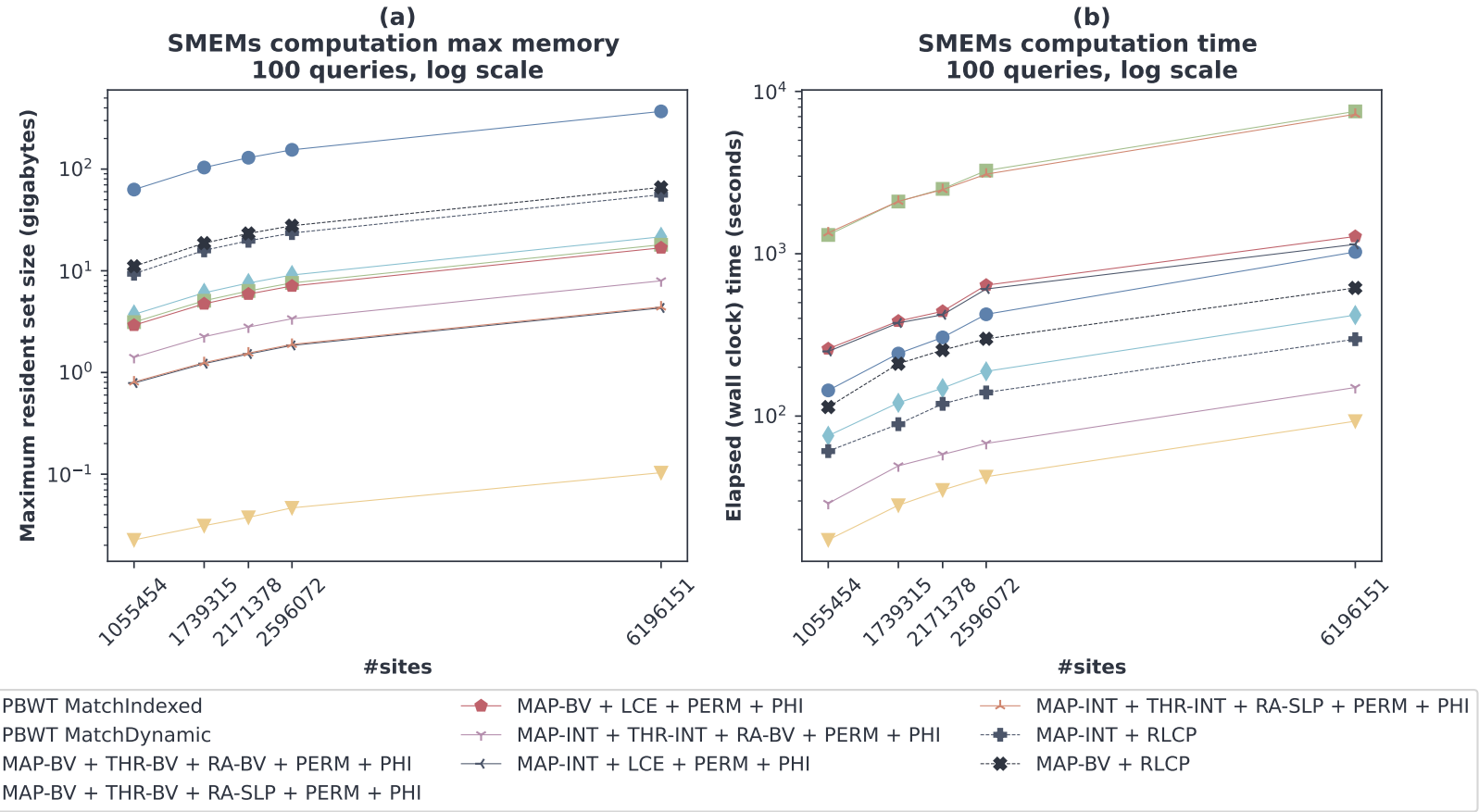
Performance costruzione strutture dati



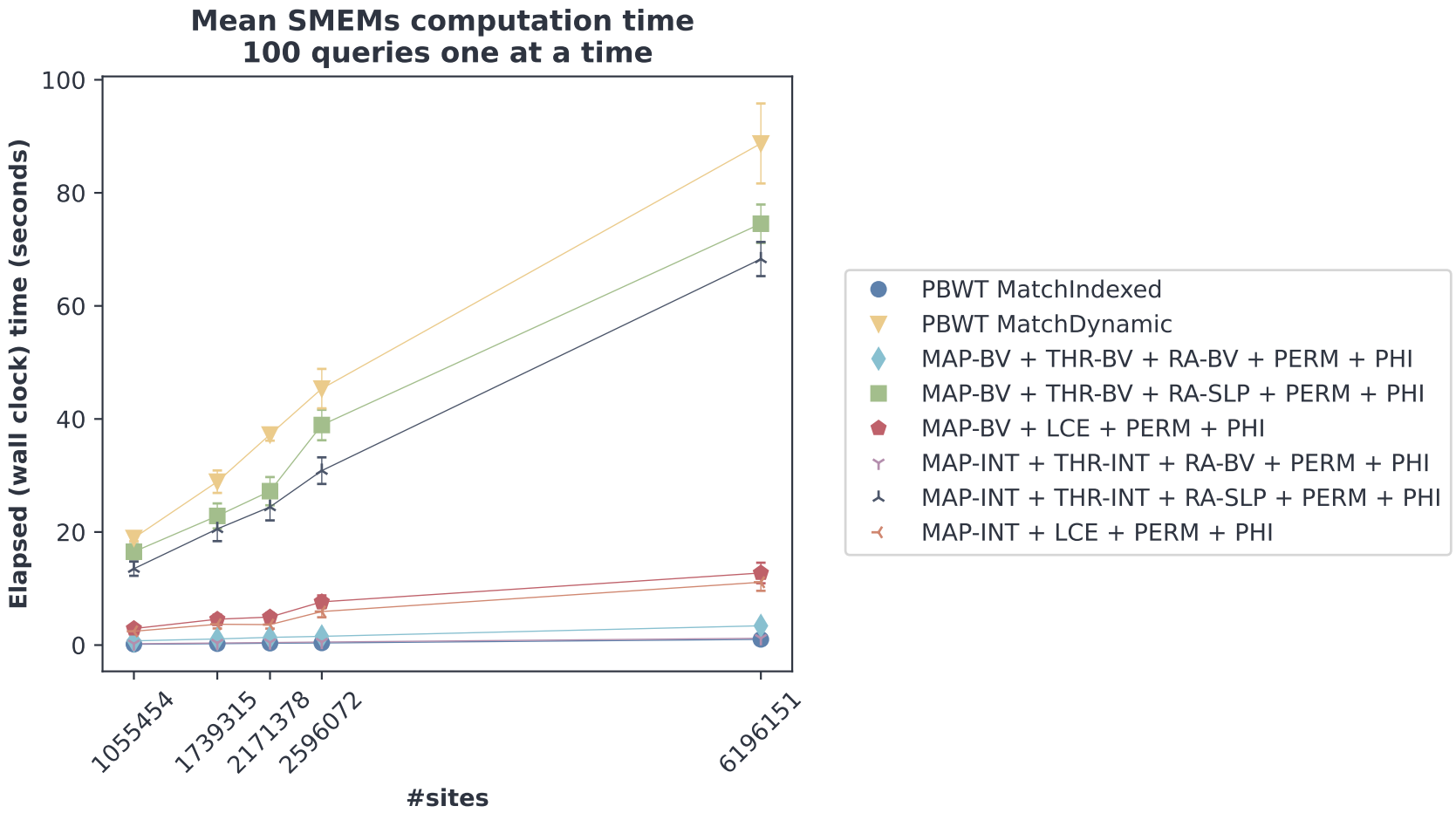
Costo in memoria delle componenti



Performance calcolo degli SMEM con 100 query



Performance calcolo degli SMEM per singole query



Outline

Considerazioni e sviluppi futuri

Alcune considerazioni

- le strutture dati e gli algoritmi proposti hanno confermato la potenzialità dell'uso di strutture run-length encoded in pangenomica
- l'obbiettivo della tesi, ovvero lo sviluppo di un algoritmo, efficiente in spazio, per il calcolo degli SMEM di un aplotipo esterno contro un pannello, è stato raggiunto con risultati molto interessanti

Sviluppi futuri

- ottimizzazioni per pannelli di query
- SMEM interni con RLPBWT
- RLPBWT con dati mancanti
- RLPBWT multiallelica
- calcolo K-SMEM con RLPBWT

Futura pubblicazione

Preprint disponibile su bioRxiv, paper in fase di review per Genome Research

Bonizzoni, Boucher, Cozzi, Gagie, Kashgouli, Köppl e Rossi:
Compressed data structures for population-scale positional Burrows–Wheeler transforms, 2022

Grazie per l'attenzione

Davide Cozzi

Relatore: *Prof.ssa Raffaella Rizzi* **Correlatore:** *Dr. Yuri Pirola*

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
Università degli Studi di Milano Bicocca

26 Ottobre 2022