

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

- 1 Introduzione e scopo della tesi
- 2 Contributo della tesi
- 3 Risultati sperimentali
- 4 Conclusioni

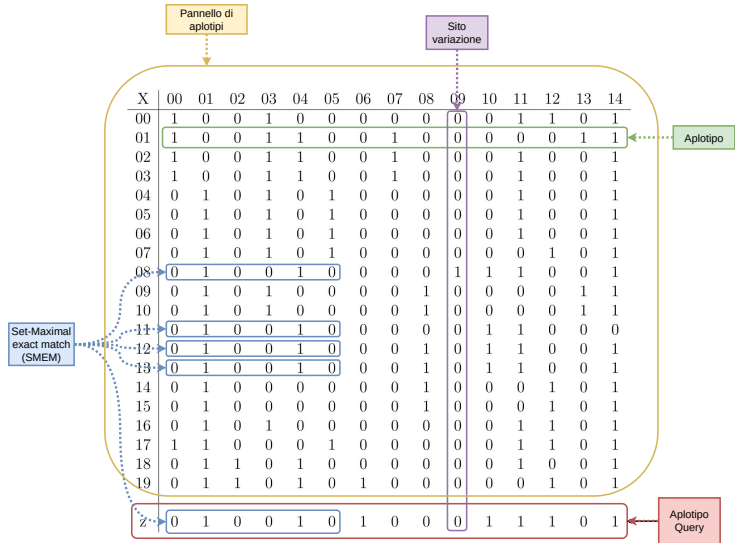
Un punto di vista per il pangenoma

Il pangenoma

- studio di un insieme di genomi provenienti da diversi individui
- Studio delle varianti geniche
- unica sequenze per multiple sequenze lineari
- grafo del pangenoma
- **pannello di aplotipi**

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

Un punto di vista per il pangenoma



Trasformata di Burrows–Wheeler posizionale

PBWT - *Durbin, Bioinformatics, 2014*

Dato pannello di M aplotipi, lunghi N siti (biallelici: $\Sigma = \{0, 1\}$), si definisce PBWT del pannello una collezione di $N + 1$ coppie di array (a_k, d_k) , $0 \leq k \leq N$, dove:

- a_k è il **prefix array** della colonna k
- d_k è il **divergence array** della colonna k

Il pannello, riordinato in ogni colonna k con a_k , è detto: matrice PBWT.

Run-length encoding

Il run-length encoding consiste nel memorizzare le *run*, ovvero sequenze massimali di caratteri uguali, come coppie:

(carattere, lunghezza della run)
 $000000 \implies (0, 6)$

Trasformata di Burrows–Wheeler posizionale

| 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 | 0 | 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 |

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

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

Trasformata di Burrows–Wheeler posizionale

| 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 |

Scopo della tesi

Calcolo degli SMEM con aplotipo esterno per Durbin:

- tempo: $\mathcal{O}(NM) + \text{Avg.}\mathcal{O}(N + c)$
- spazio: $\mathcal{O}(NM) \Rightarrow 13NM$ byte

La PBWT tendo ad avere simboli uguali in posizioni consecutive all'interno di ogni sua colonna, a causa del riordinamento e della natura biologica del dato.

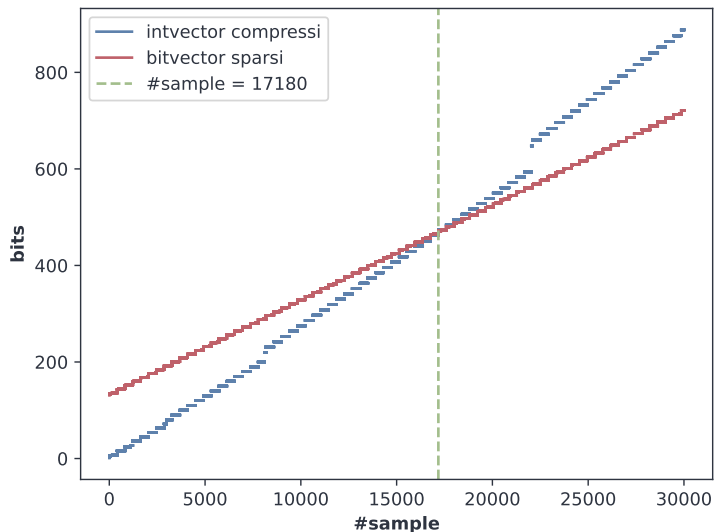
Lo scopo di questa tesi è stato quello di creare una variante run-length encoded della PBWT (RLPBWT) che permettesse, in modo efficiente dal punto di vista della memoria richiesta, il calcolo degli SMEM con aplotipo esterno.

Le componenti

Componenti innovative derivate dallo studio della RLBWT in ottica PBWT

- mapping tra una colonna e la successiva nella PBWT e threshold:
 - bitvector sparsi, con rank in $\mathcal{O}\left(\log\left(\frac{M}{\rho}\right)\right)$: MAP-BV e THR-BV
 - intvector compressi, con rank in $\mathcal{O}(\log(\rho))$: MAP-INT e THR-INT
- random access:
 - bitvector, in $\mathcal{O}(1)$: RA-BV
 - SLP, in $\mathcal{O}(\log(NM))$: RA-SLP
- LCE query con SLP, in $\mathcal{O}(\log(NM))$: LCE
- prefix array sample: PERM
- struttura per le funzioni φ e φ^{-1} : PHI
- reverse longest common prefix: RLCP

Qualche confronto in spazio



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

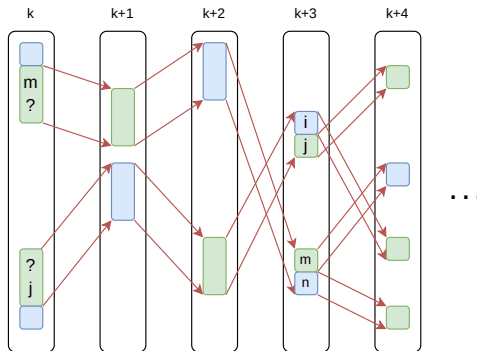
- ① usare le threshold e il random access al pannello (ispirandosi a MONI - Rossi et al., *Journal of Computational Biology*, 2022)
- ② usare le LCE query (ispirandosi a PHONI - Boucher et al., *Data Compression Conference (DCC)*, 2021)

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}



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

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



Sperimentazione e dati

Implementazione e sperimentazione

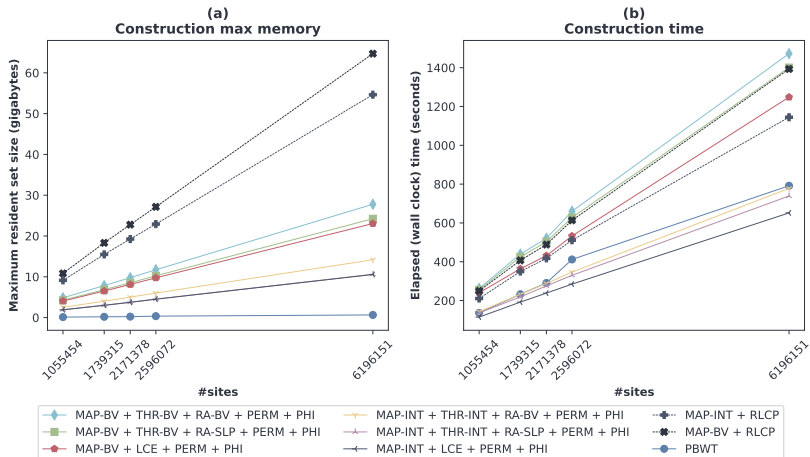
La sperimentazione, orchestrata tramite `snakemake`, è stata effettuata su una macchina con processore Intel Xeon E5-2640 V4 (2,40GHz), 756GB di RAM, 768GB di swap e sistema operativo Ubuntu 20.04.4 LTS.

Si sono confrontate l'implementazione in C++ della RLPBWT e l'implementazione in C ufficiale della PBWT.

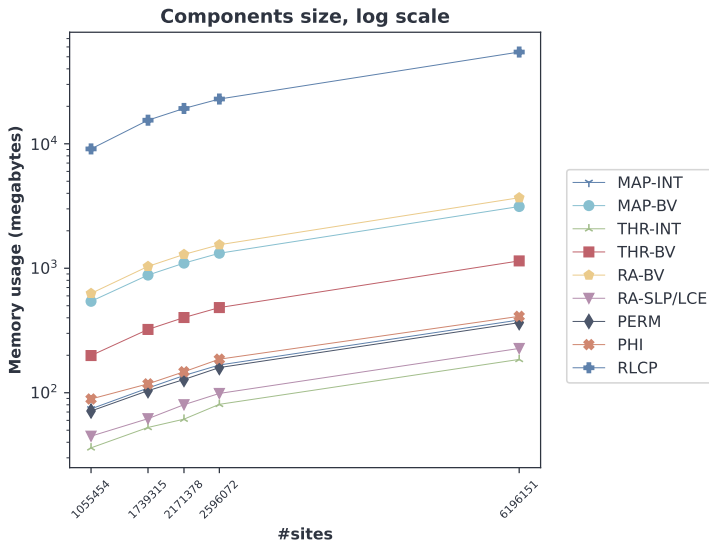
Pannelli del 1000 Genome Project con 4908 sample, avendone estratti 100 come query.

| 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 |

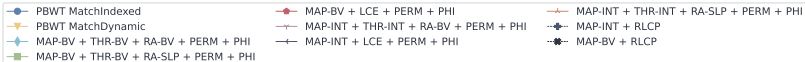
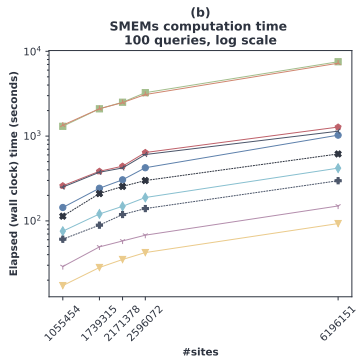
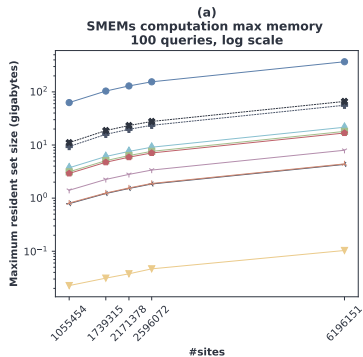
Performance costruzione strutture dati



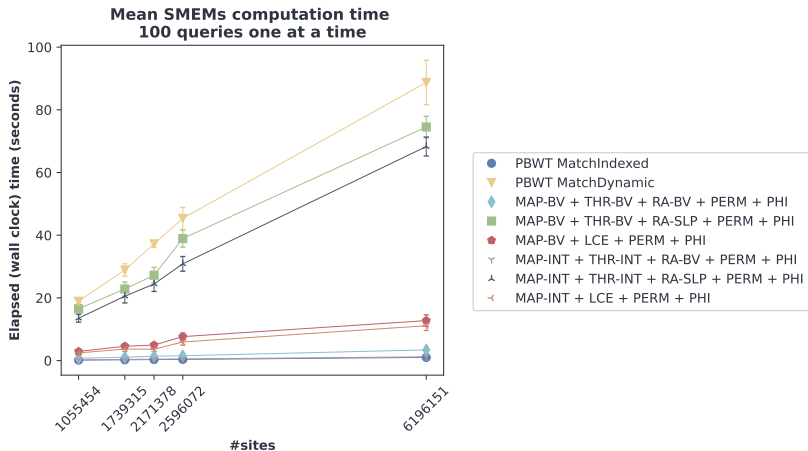
Costo in memoria delle componenti



Performance calcolo degli SMEM con 100 query



Performance calcolo degli SMEM per singole query



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'obiettivo 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

Ulteriori dettagli

Bonizzoni, Boucher, Cozzi, Gaggie, Kashgouli, Köppl e Rossi:

Compressed data structures for population-scale positional Burrows–Wheeler transforms,
bioRxiv (preprint), 2022

Grazie per l'attenzione



DALHOUSIE
UNIVERSITY

