Solving the Minimal Positional Substring Cover problem in sublinear space

Paola Bonizzoni[1] Christina Boucher[2] **Davide Cozzi**[1]
Travis Gagie[3] Yuri Pirola[1]

[1]University of Milano-Bicocca, Italy

[2]University of Florida, USA

[3]Dalhousie University, Canada

CPM 2024 – June 25, 2024 – Fukuoka

Context

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

Haplotype

Set-Maximal exact match (SMEM)

External haplotype

Our contribution

The haplotype threading problem

The haplotype threading problem is to represent a query haplotype by using one or more substrings derived from the haplotypes within a reference panel.

We solved in sublinear space

- Minimal Positional Substring Cover (MPSC) problem
- the leftmost MPSC problem
- the rightmost MPSC problem
- the length-maximal MPSC problem
- the k-MPSC problem

Our contribution

To solve these problems, we exploit the use of the μ-PBWT (*Bioinformatics, 2023*), augmenting it with the possibility to compute Matching Statistics (MS) and Set-Maximal Exact Matches (SMEMs) that involve at least k rows.

μ-PBWT is able to compute MS and SMEMs in sublinear space.

Positional Burrows–Wheeler Transform

Divergence array (DA)

run

i	d_5	a_5	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
00	05	14	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
01	00	15	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1
02	01	17	1	1	0	0	0	1	0	0	0	0	0	1	1	0	1
03	04	00	1	0	0	1	0	0	0	0	0	0	0	1	1	0	1
04	02	04	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
05	00	05	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
06	00	06	0	1	0	1	0	1	0	0	0	0	0	1	0	0	1
07	00	07	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1
08	00	09	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
09	00	10	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1
10	00	16	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1
11	05	08	0	1	0	0	1	0	0	0	0	1	1	1	0	0	1
12	00	11	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0
13	00	12	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
14	00	13	0	1	0	0	1	0	0	0	1	0	1	1	0	0	1
15	03	18	0	1	1	0	1	0	0	0	0	0	0	1	0	0	1
16	00	19	0	1	1	0	1	0	1	0	0	0	0	0	1	0	1
17	04	01	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1
18	00	02	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1
19	00	03	1	0	0	1	1	0	0	1	0	0	0	1	0	0	1

Prefix array (PA)

Matching statistics are able to compute SMEMs

Matching statistics

Given a panel $M = \{m_0, \ldots, m_{h-1}\}$, s.t. $|m_i| = w$, and an external haplotype P ($|P| = w$), we define **matching statistics** of P against M as an array MS of pair (row, len), $|\text{MS}| = w$, such that:

- $m_{\text{MS}[i].\text{row}}[i - \text{MS}[i].\text{len} + 1, i] = P[i - \text{MS}[i].\text{len} + 1, i]$, so we have a MS[$i$].len-long match between P and MS[i].row to the i-th column
- $P[i - \text{MS}[i].\text{len}, i]$ is not a suffix of any subset of row of M

Set-maximal exact match (SMEM)

A triple (A, b, e) such that:

- $m_i[b : e] = P[b : e]$ for each $i \in A$ ((A, b, e) is a set match)
- $m_i[b - 1] \neq P[b - 1]$ and $m_i[e + 1] \neq P[e + 1]$
- $\nexists\, j \in \{0, 1, \ldots h - 1\} \backslash A$ such that there is a set match from P to x_j that includes the interval $[b : e]$

Matching statistics are able to compute SMEMs

Compute SMEMs from MS

Given a matching statistics array MS, $P[i-\ell+1, i]$ shares a ℓ-long SMEM with MS$[i]$.row iff:

- MS$[i]$.len $= \ell$
- $i = w - 1 \vee$ MS$[i]$.len $\geq$ MS$[i+1]$.len

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0
2	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
3	1	0	1	0	1	0	0	1	0	0	0	1	1	0	0
4	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0
5	0	1	0	1	0	0	0	0	1	1	0	1	0	0	1
6	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0

MS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	1	1	1	0	1	0	1	1	0	0	1	1	1	0	0
row	6	4	4	4	4	3	1	1	1	1	6	6	3	3	3
len	1	2	3	4	5	4	5	6	7	8	5	6	2	3	4

Minimal Positional Substring Cover Problem

Positional substring

A positional substring of a string X is a triplet (i, j, X) with $1 \leq i, j \leq |X|$ and we say that the substring corresponding to (i, j, X) is $X[i..j]$.

Minimal Positional Substring Cover (MPSC) problem

Given a set M of strings of length w (i.e., panel), a *positional substring cover* of a w-length string P by M is a set C of positional substrings s.t.:

- each $\ell \in [1, w]$ of P is covered by a $(i, j, X) \in C$ (i.e., $i \leq \ell \leq j$)
- the substring corresponding to each $(i, j, X) \in C$ is contained in P
- the substring corresponding to each $(i, j, X) \in C$ is contained in at least one string of M

We are then interested in a positional substring cover which is also minimal, i.e. with the smallest size over all positional substring covers of P by S.

Minimal Positional Substring Cover Problem

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0
2	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
3	1	0	1	0	1	0	0	1	0	0	0	1	1	0	0
4	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0
5	0	1	0	1	0	0	0	0	1	1	0	1	0	0	1
6	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0

P	1	1	1	0	1	0	1	1	0	0	1	1	1	0	0

MPSC problem variants

Some variants proposed in literature

- Find a *leftmost MPSC* C of P by M, i.e. a MPSC of P by M such that any i-th substring in C starts at least as early as the i-th substring of every other MPSC of P by M
- Find a *rightmost MPSC* C of P by M, i.e. a MPSC of P by M such that any i-th substring in C ends at least as late as the i-th substring of every other MPSC of P by M
- Find a *length-maximal MPSC* of P by M, i.e. the MPSC that has the largest length out of all MPSCs of P by M

k-Minimal Positional Substring Cover (MPSC) problem

Every MPSC variant can be extended to consider at least k strings of M.

Our approach solves left/rightmost MPSC problem

M	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0
2	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
3	1	0	1	0	1	0	0	1	0	0	0	1	1	0	0
4	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0
5	0	1	0	1	0	0	0	0	1	1	0	1	0	0	1
6	1	0	1	0	1	1	1	1	0	0	1	1	0	0	0

MS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	1	1	1	0	1	0	1	1	0	0	1	1	1	0	0
row	6	4	4	4	4	3	1	1	1	1	6	6	3	3	3
len	1	2	3	4	5	4	5	6	7	8	5	6	2	3	4
j/j'	$\leftarrow$	2	$\leftarrow$	$\leftarrow$	$\leftarrow$	6	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	11	$\leftarrow$	$\leftarrow$	$\leftarrow$	15

Figure: How to compute leftmost MPSC by using Matching Statistics

Our approach solves length-maximal MPSC problem

Length-maximal MPSC problem

Sanaullah et al. (*Genome Research, 2023*) showed the length-maximal MPSC problem can be solved using a leftmost MPSC, a rightmost MPSC, and the set of SMEMs and we showed that our approach is able to compute all these "ingredients".

Extending to k-MPSC

k-Minimal Positional Substring Cover (MPSC) problem

Every MPSC variant can be extended to consider at least k strings of M.

Extension

We extend the notion of MS and SMEMs obtaining:

- k-MS: MS s.t. each left-maximal match represented in the MS array involves at least k rows in the input panel
- k-SMEMs: SMEMs that involve at least k rows in the input panel

μ-PBWT was able to compute MS and SMEMs in sublinear space and it is extended also to compute k-MS and k-SMEMs in sublinear space.

Results on 1000 Genomes Project data

Panels details

- panels from phase 3 of **1000 Genomes Project**
- fixed number of haplotypes (5008) and variable number of sites
- 30 haplotypes as queries $\implies$ panels with 4978 haplotypes
- comparison against Sanaullah et al. MPSC implementation (*Genome Research, 2023*)
- tested different k values ($k = 1$ for classical MS and SMEMs)

Chromosome	**#Sites**	**Avg. runs**
`chr18`	2,171,378	11
`chr2`	6,786,300	11

Results on 1000 Genomes Project data

Chr.	Task	k	Wall Clock Time (seconds)		Max Memory usage (GB)	
			μ-PBWT	k-MPSC	μ-PBWT	k-MPSC
18	*Index*	1	**425**	5750	**3.26**	168.58
		50	**697**	5750	**3.36**	168.58
		200	**807**	5750	**3.36**	168.58
	Querying	1	**70**	128	**1.92**	171.19
		50	739	**229**	**2.13**	171.30
		200	1740	**119**	**2.74**	171.22
2	*Index*	1	1381	-	6.45	-
		50	1643	-	6.62	-
		200	2798	-	6.62	-
	Querying	1	254	-	5.87	-
		50	1687	-	6.57	-
		200	3563	-	8.45	-

Conclusions

Take home message

We addressed the theoretical aspects of the haplotype threading problem via the MPSC problem, primarily aiming to offer practical solutions capable of scaling to large biological datasets. Our approach opens up unprecedented opportunities for comprehensive genetic studies and exploration on a large scale.

Future developments

- Optimize the implementation and perform experiments on bigger datasets such as UK Biobank data.
- Integration of μ-PBWT in GLIMPSE (*Nature Genetics, 2021*), a state-of-the-art haplotype phasing and genotype imputation tool for low-coverage sequencing datasets using haplotype threading.

Thank you for your attention!