

Efficient sequence similarity searches with strobemers and applications in read mapping

Kristoffer Sahlin

Department of Mathematics, Science for Life Laboratory
Stockholm University

Finding similarity between sequences

- Used in:
 - Sequence {mapping, clustering, assembly}
 - SNP and SV detection
 - Storage and retrieval
- Larger datasets and more redundant databases
- I will discuss read mapping to reference(s)

Finding similarity between sequences

- Used in:
 - Sequence {mapping, clustering, assembly}
 - SNP and SV detection
 - Storage and retrieval
- Larger datasets and more redundant databases
- I will discuss read mapping to reference(s)

Read mapping

- Lossless approaches (guaranteed best match):
 - Pairwise alignment (slow)
 - Efficient search schemes for up to X (≤ 4) errors.
 - Pigeonhole principle with ILP
 - Lossy approaches (seed–filter–extend)
 - FM-index: BWA, Bowtie2, ...
 - Hash table: BLAT, minimap2, Winnowmap2, ...
- ← This talk

Hash-based seed-filter-extend

Indexing

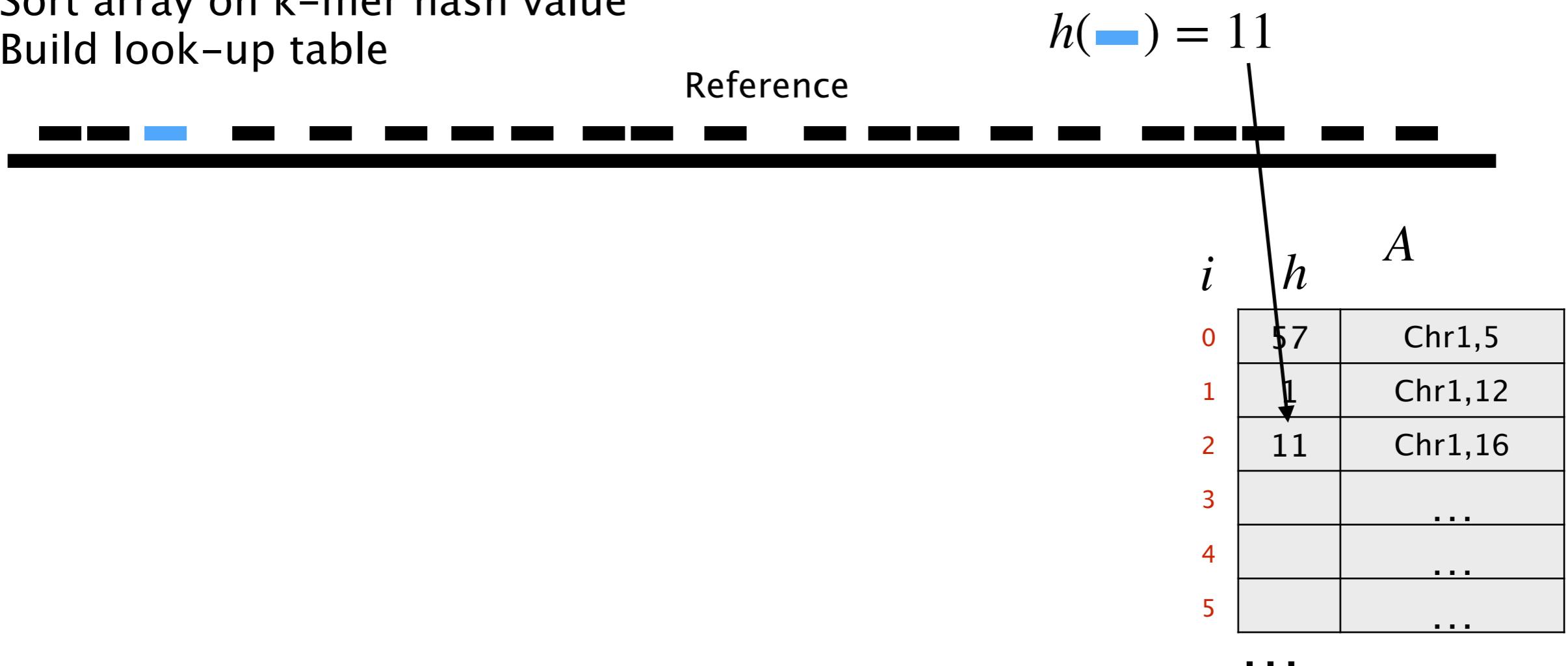
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Hash-based seed-filter-extend

Indexing

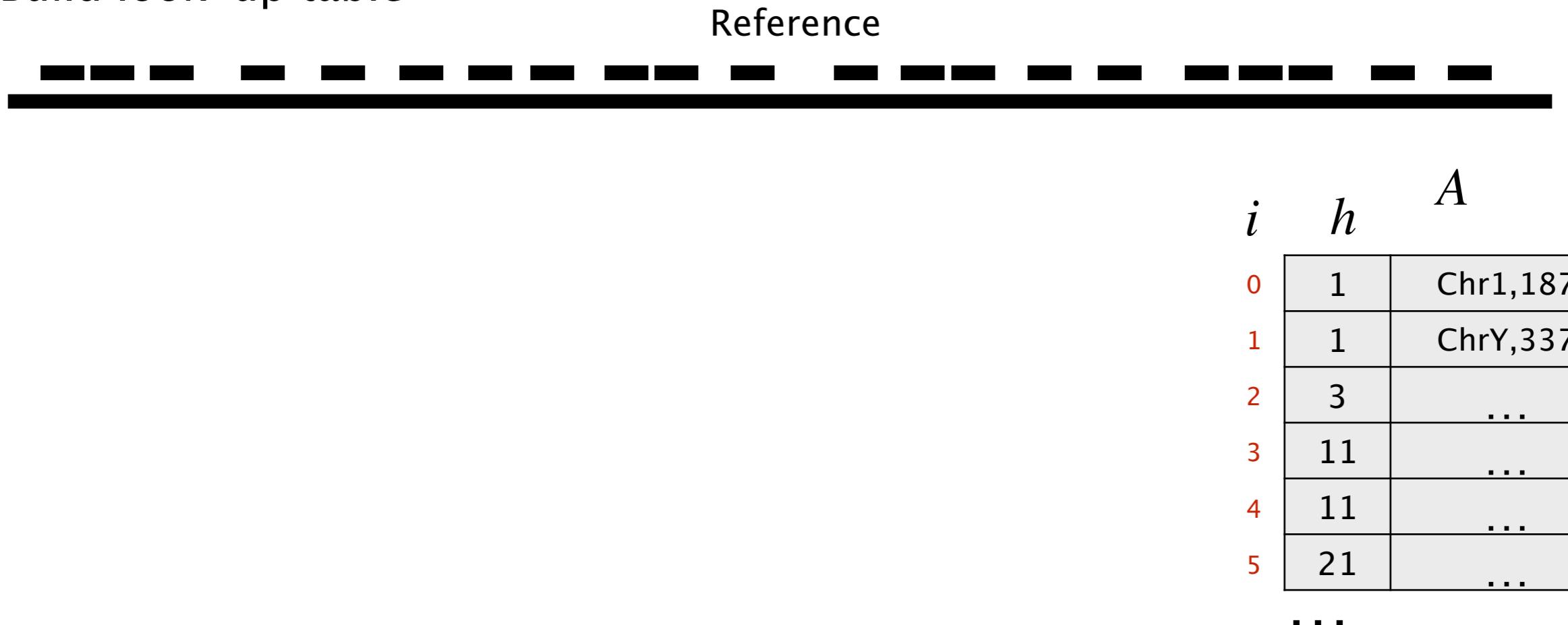
1. Extract k-mers, hash, and put in array
 2. Sort array on k-mer hash value
 3. Build look-up table



Hash-based seed-filter-extend

Indexing

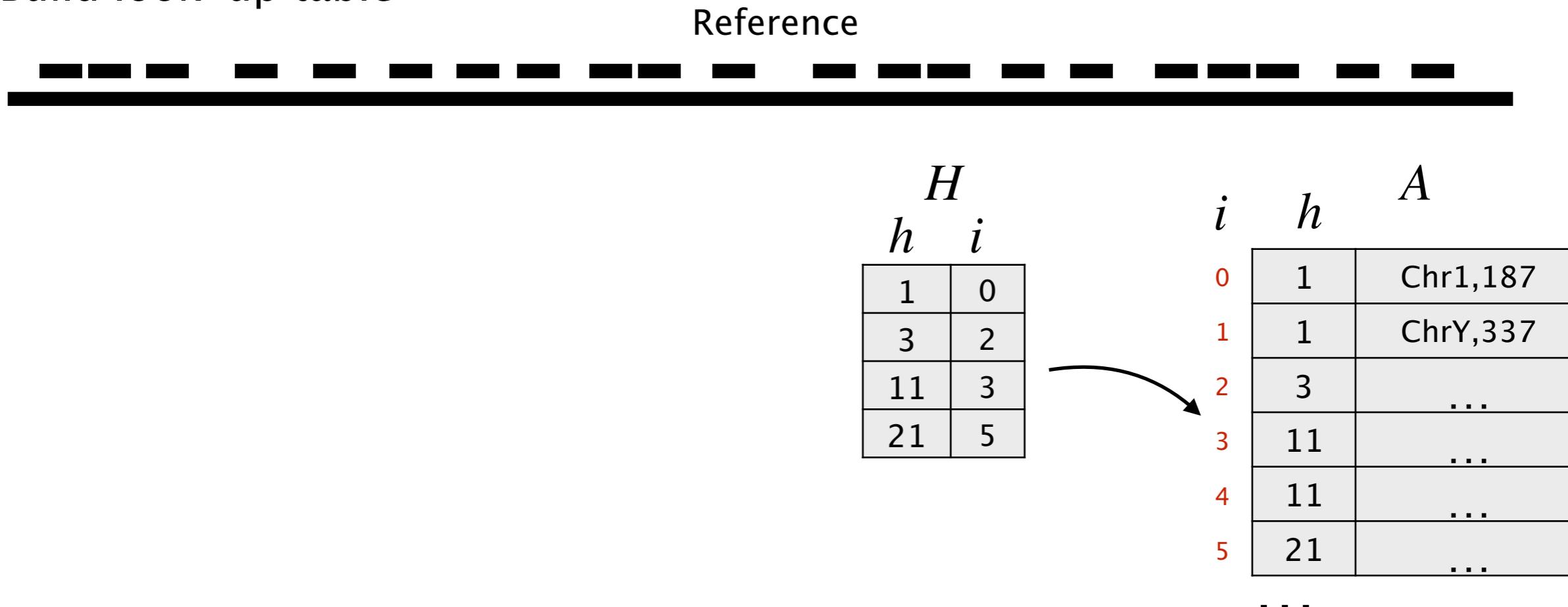
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Hash-based seed-filter-extend

Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Hash-based seed-filter-extend

Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



$$h(\text{---}) = 11$$

Reference

H	
h	i
1	0
3	2
11	3
21	5

i	h	A
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

Hash-based seed-filter-extend

Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment



$$h(\text{---}) = 11$$

Reference

H	
h	i
1	0
3	2
11	3
21	5

i	h	A
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

Hash-based seed-filter-extend

Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



H		i	A
h	i		
1	0	0	Chr1,187
3	2	1	ChrY,337
11	3	2	...
21	5	3	...
		4	11
		5	11
	

Hash-based seed-filter-extend

Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Mapping

1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



Reference

H	i
h	
1	0
3	2
11	3
21	5

i	h	A
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

Hash-based seed-filter-extend

Indexing

1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Mapping

This talk

1. Seed: Find shared k-mers subsequences
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment

Query



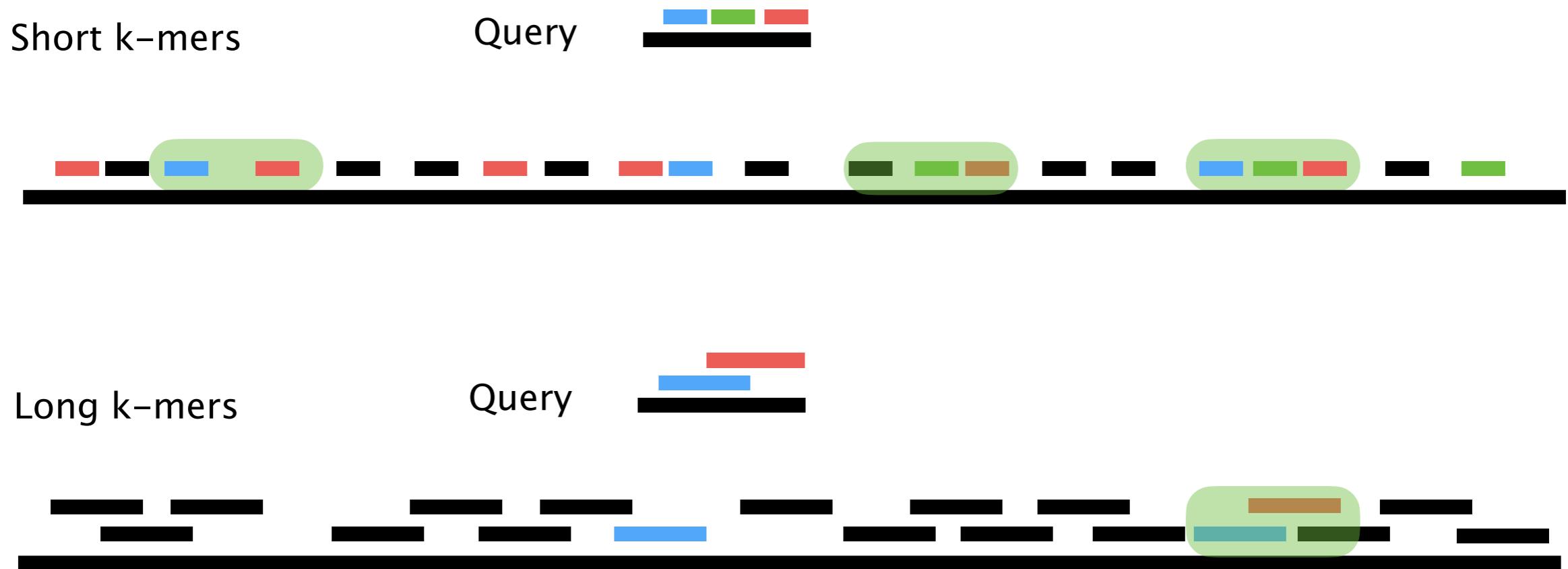
Reference

H	
h	i
1	0
3	2
11	3
21	5

i	h	A
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
...		

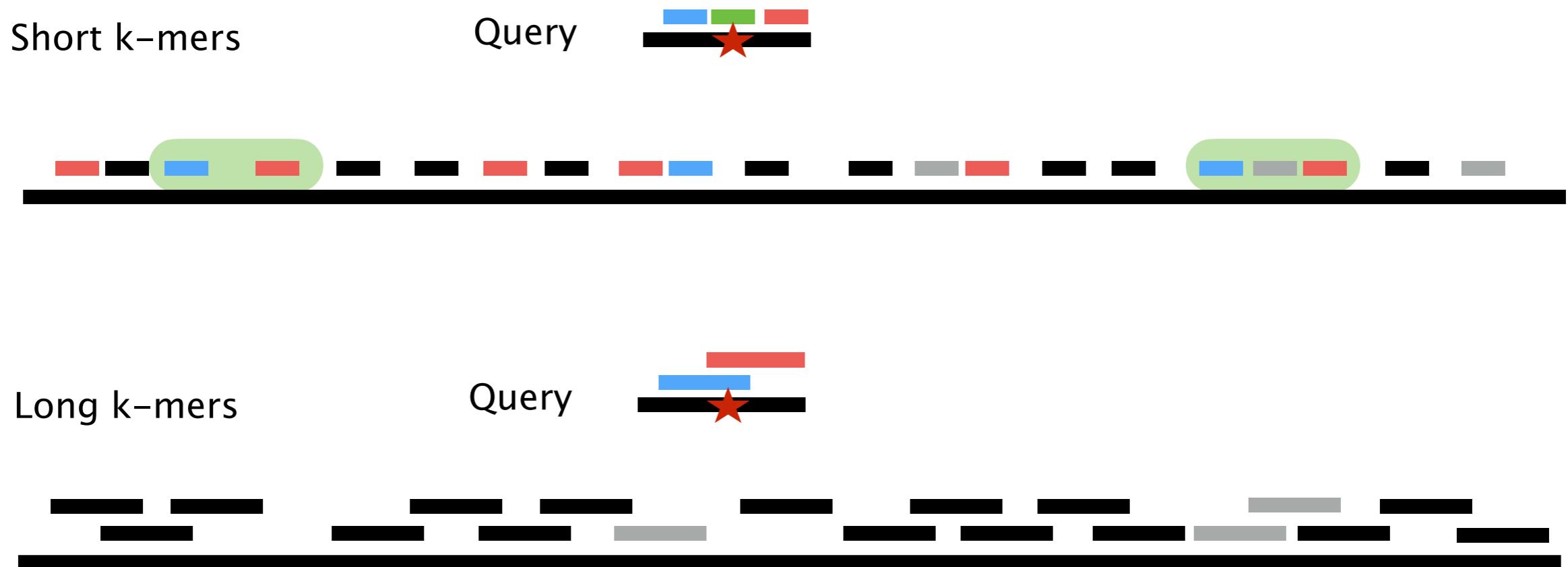
Sensitivity-specificity trade-off

- Short k-mers: high sensitivity, low specificity
- Long k-mers: low sensitivity, high specificity



Sensitivity-specificity trade-off

- Short k-mers: high sensitivity, low specificity
- Long k-mers: low sensitivity, high specificity



Approximate-matching (fuzzy) seeds

- Spaced k-mers¹
- Vector seeds²
- Permutation techniques^{3,4}
- Covering template families⁵
- SimHash³ (BLEND⁶)
- TensorSketch⁷
- SubSeqHash⁸
- Neighbouring minimizers⁹
- k-min-mers¹⁰
- Strobemers¹¹



ACTC GGTA CTAG
GACT TGGT CCTA
CGAC CTGG ACCT
ACGA TCTG TACC AGAC
CACG CTCT GTAC TAGA
CACCGACTCTGGTACCTAGAC
||| ||| ||| ||| ||| |||
CACGG**C**TC**A**GGT-**C**CTAGAC

¹ Ma et al., 2002

² Brejová et al., 2003

³ Charikar, 2002

⁴ Lederman, 2013

⁵ Giladi et al., 2010

⁶ Firtina et al., 2021

⁷ Joudaki et al., 2020

⁸ Li et al., 2023

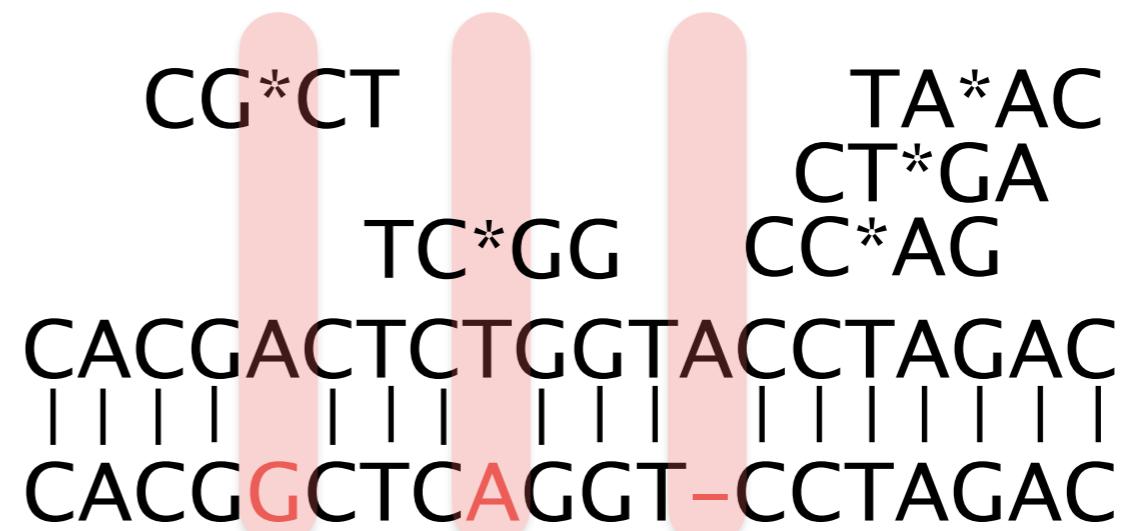
⁹ Chin and Khalak, 2019

¹⁰ Ekim, Berger, Chikhi, 2021

¹¹ Sahlin, 2021

Approximate-matching (fuzzy) seeds

- Spaced k-mers¹
 - Vector seeds²
 - Permutation techniques^{3,4}
 - Covering template families⁵
 - SimHash³ (BLEND⁶)
 - TensorSketch⁷
 - SubSeqHash⁸
 - Neighbouring minimizers⁹
 - k-min-mers¹⁰
 - Strobemers¹¹
- + Match over substitutions
 - Does not handle indels



¹ Ma et al., 2002

² Brejová et al., 2003

³ Charikar, 2002

⁴ Lederman, 2013

⁵ Giladi et al., 2010

⁶ Firtina et al., 2021

⁷ Joudaki et al., 2020

⁸ Li et al., 2023

⁹ Chin and Khalak, 2019

¹⁰ Ekim, Berger, Chikhi, 2021

¹¹ Sahlin, 2021

Approximate-matching (fuzzy) seeds

- Spaced k-mers¹
 - Vector seeds²
 - Permutation techniques^{3,4}
 - Covering template families⁵
 - SimHash³ (BLEND⁶)
 - TensorSketch⁷
 - SubSeqHash⁸
 - **Neighbouring minimizers⁹**
 - k-min-mers¹⁰
 - Strobemers¹¹

ACTC GGT A CTAG
GA CT TG GT CCTA
CG AC CT GG AC CT
AC GA TCTG TACC AG AC
CAC G CTCT GTAC TAGA
CAC G ACTCTGGTAC CCTA GAC
||| ||| ||| ||| |||
CAC G ACTCTGG -- CCTA GAC

10 Ma et al., 2002

¹¹ Brejová et al., 2003

12 Charikar, 2002

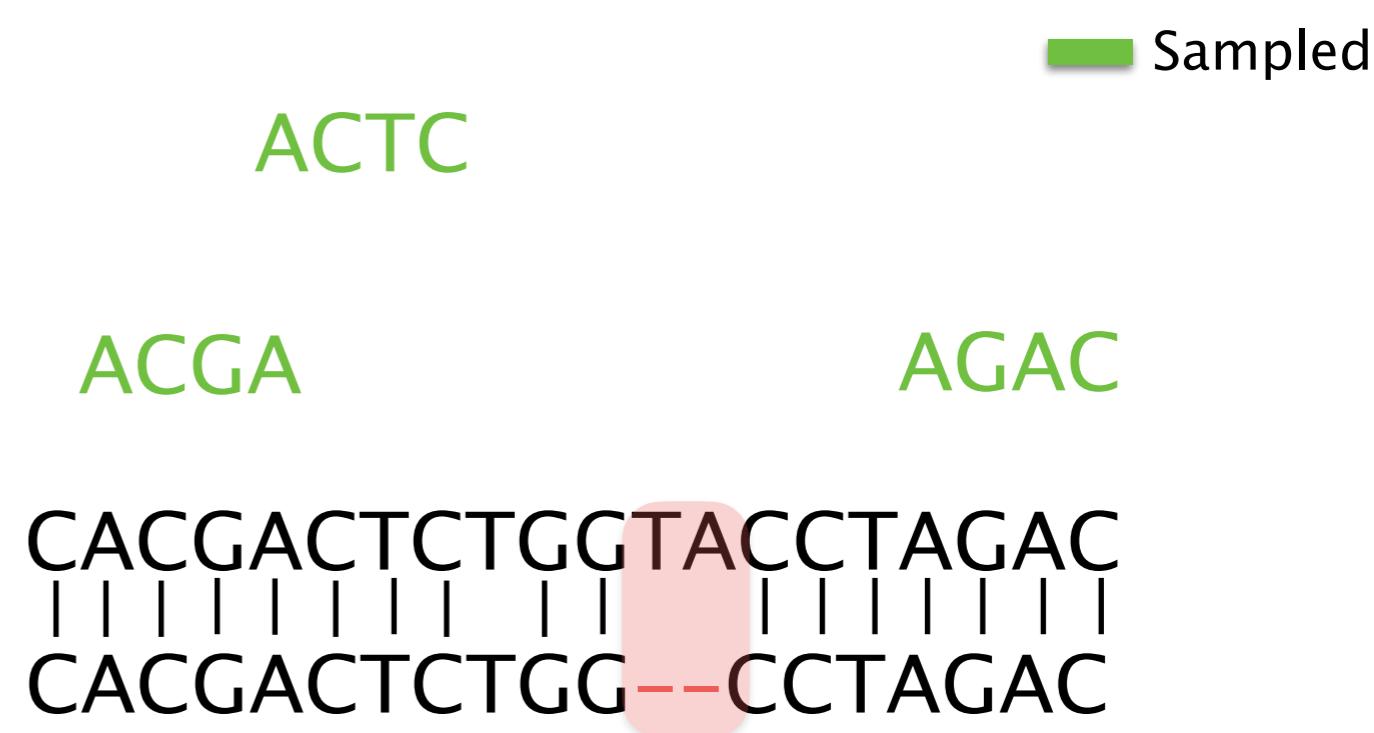
¹³ Lederman, 2013

¹⁴ Giladi et al., 2010

¹⁵ Firtina et al., 2021

Approximate-matching (fuzzy) seeds

- Spaced k-mers¹
- Vector seeds²
- Permutation techniques^{3,4}
- Covering template families⁵
- SimHash³ (BLEND⁶)
- TensorSketch⁷
- SubSeqHash⁸
- **Neighbouring minimizers⁹**
- k-min-mers¹⁰
- Strobemers¹¹



¹⁰ Ma et al., 2002

¹¹ Brejová et al., 2003

¹² Charikar, 2002

¹³ Lederman, 2013

¹⁴ Giladi et al., 2010

¹⁵ Firtina et al., 2021

× Joudaki et al., 2020

✗ Li et al., 2023

✗ Chin and Khalak, 2019

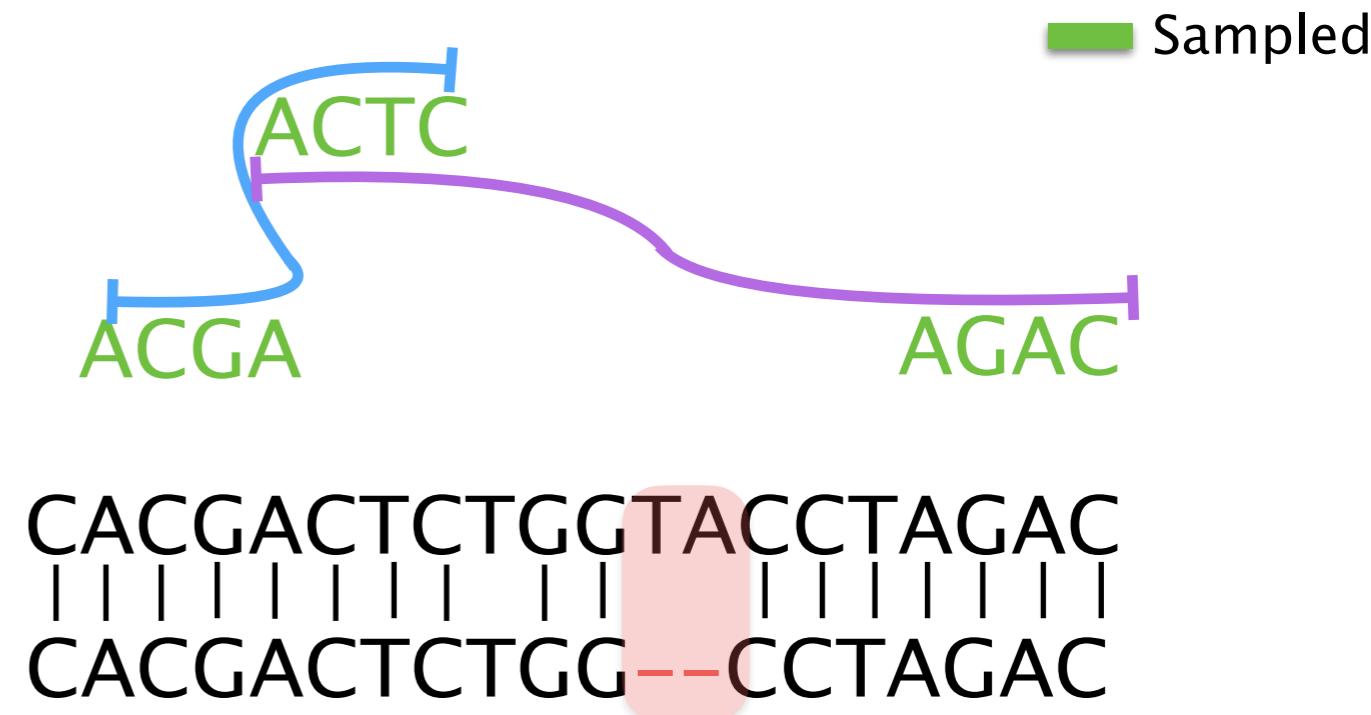
¹¹ Ekim, Berger, Chikhi, 2021

✗ Sahlin, 2021

Approximate-matching (fuzzy) seeds

- Spaced k-mers¹
- Vector seeds²
- Permutation techniques^{3,4}
- Covering template families⁵
- SimHash³ (BLEND⁶)
- TensorSketch⁷
- SubSeqHash⁸
- **Neighbouring minimizers⁹**
- k-min-mers¹⁰
- Strobemers¹¹

- + Match over subs. and indels
- “k-mer redundancy”



¹⁰ Ma et al., 2002

¹¹ Brejová et al., 2003

¹² Charikar, 2002

¹³ Lederman, 2013

¹⁴ Giladi et al., 2010

¹⁵ Firtina et al., 2021

× Joudaki et al., 2020

✗ Li et al., 2023

✗ Chin and Khalak, 2019

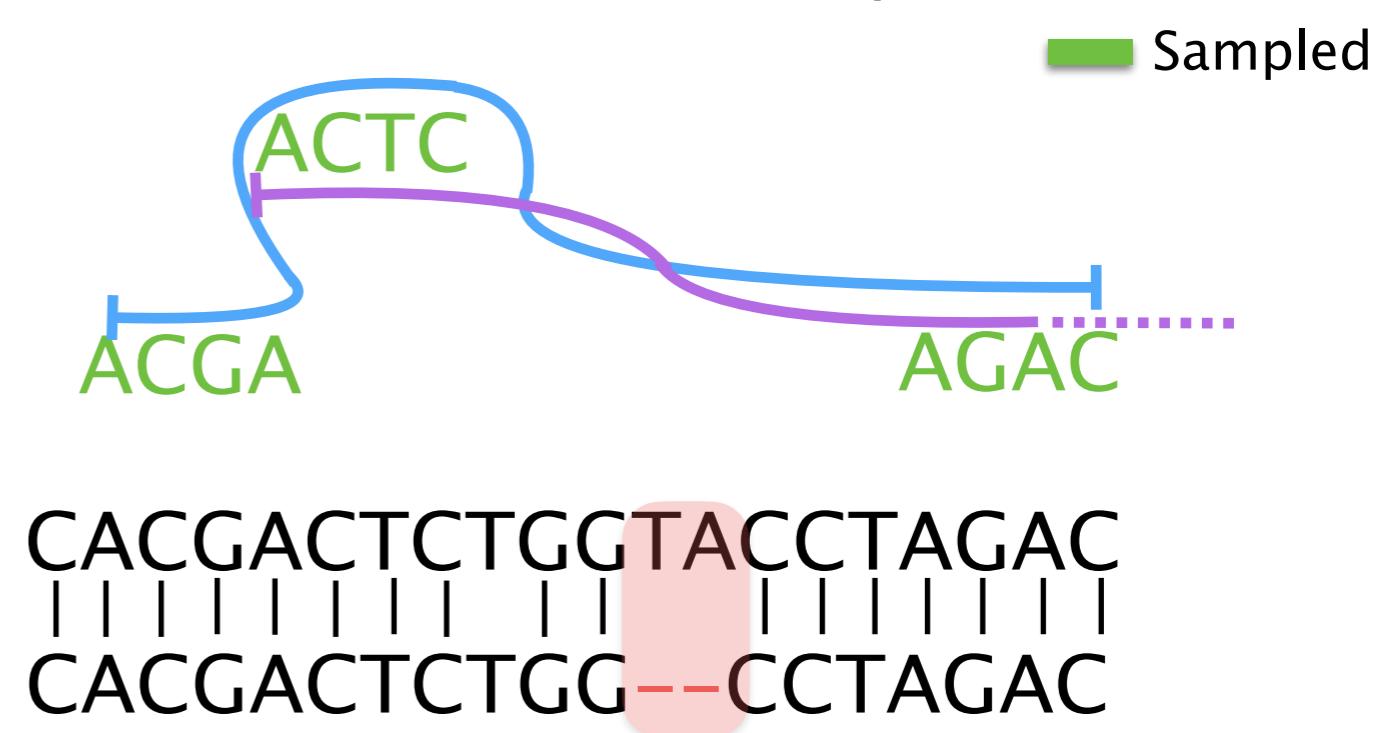
¹¹ Ekim, Berger, Chikhi, 2021

✗ Sahlin, 2021

Approximate-matching (fuzzy) seeds

- Spaced k-mers¹
- Vector seeds²
- Permutation techniques^{3,4}
- Covering template families⁵
- SimHash³ (BLEND⁶)
- TensorSketch⁷
- SubSeqHash⁸
- Neighbouring minimizers⁹
- **k-min-mers**¹⁰
- Strobemers¹¹

- + Match over subs. and indels
- “k-mer redundancy”



¹⁰ Ma et al., 2002

¹¹ Brejová et al., 2003

¹² Charikar, 2002

¹³ Lederman, 2013

¹⁴ Giladi et al., 2010

¹⁵ Firtina et al., 2021

× Joudaki et al., 2020

✗ Li et al., 2023

✗ Chin and Khalak, 2019

¹¹ Ekim, Berger, Chikhi, 2021

✗ Sahlin, 2021

Strobemers

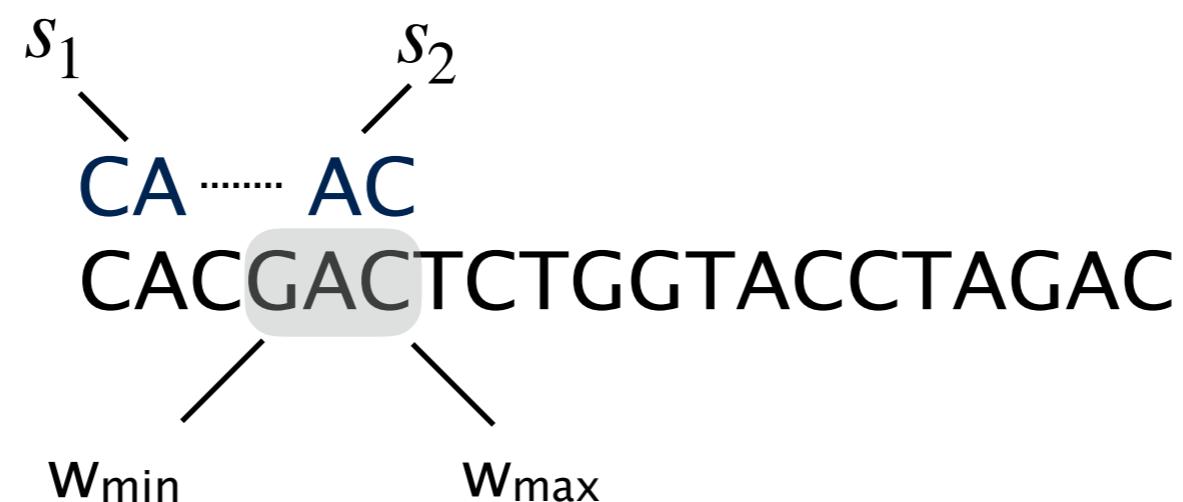
- Strobemers: n k-mers s_1, s_2, \dots, s_n (strobos) of length ℓ
 - s_1 is fixed
 - s_2, \dots, s_n chosen from downstream window

Strobemers

- Strobemers: n k-mers s_1, s_2, \dots, s_n (strobos) of length ℓ
 - s_1 is fixed
 - s_2, \dots, s_n chosen from downstream window

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

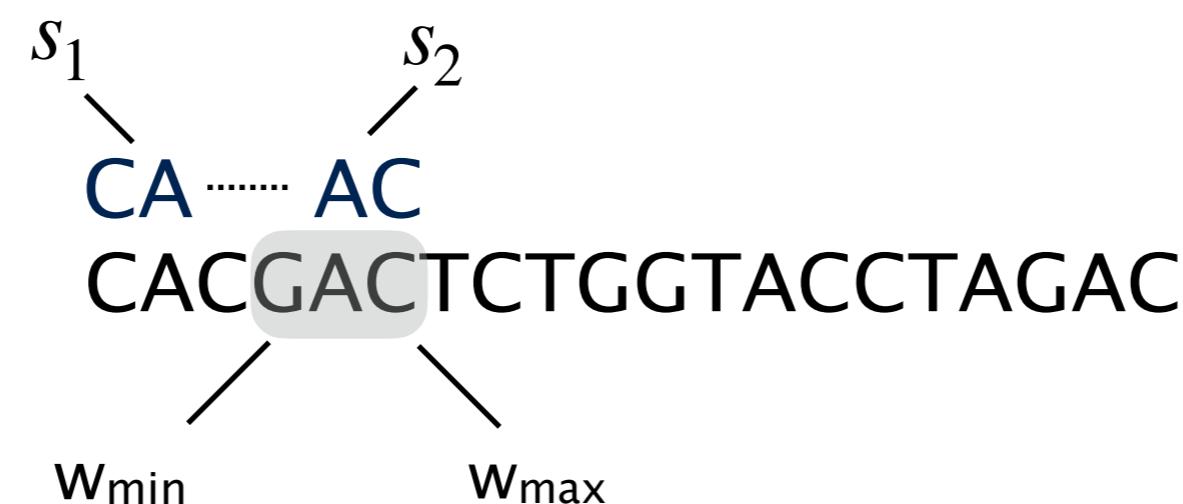


Strobemers

- Strobemers: n k-mers s_1, s_2, \dots, s_n (strobos) of length ℓ
 - s_1 is fixed
 - s_2, \dots, s_n chosen from downstream window **with a method**
 - **Methods:** `minstrokes`, `randstrokes`, `hybridstrokes`

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)



Randstrobes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

← Integer
Hash function

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

s_1
CA
CACGACTCTGGTACCTAGAC

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$

$$h(TT) = 15$$

$$q=5$$

CA
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

...

$$h(TT) = 15$$

$$q=5$$

CA
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

$$\begin{aligned} h(GA) &= 8 \\ h(AC) &= 1 \\ h(CT) &= 7 \end{aligned}$$

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

$$\begin{aligned}h(AA) &= 0 \\h(AC) &= 1\end{aligned}$$

$$\dots \\h(TT) = 15$$

$$q=5$$

CA
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

$$\begin{aligned}h(GA) &= 8 & (4 + 8) \% 5 &= 2 \\h(AC) &= 1 & (4 + 1) \% 5 &= 0 \\h(CT) &= 7 & (4 + 7) \% 5 &= 1\end{aligned}$$

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$

$$h(TT) = 15$$

$$q=5$$

CA.....AC
CACGACTGTGGTACCTAGAC

$$h(CA) = 4$$

$$\begin{aligned} h(GA) &= 8 & (4 + 8) \% 5 &= 2 \\ h(AC) &= 1 & (4 + 1) \% 5 &= 0 \\ h(CT) &= 7 & (4 + 7) \% 5 &= 1 \end{aligned}$$

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$

$$h(TT) = 15$$

$$q=5$$

AC
CA AC
CACGACTGTGGTACCTAGAC

$$h(AC) = 1$$

$$\begin{aligned} h(AC) &= 1 & (1 + 1) \% 5 &= 2 \\ h(CT) &= 7 & (1 + 7) \% 5 &= 3 \\ h(TG) &= 14 & (1 + 14) \% 5 &= 0 \end{aligned}$$

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

$$\begin{aligned} h(AA) &= 0 \\ h(AC) &= 1 \end{aligned}$$

$$\dots$$

$$h(TT) = 15$$

$$q=5$$

AC TG
CA AC
CACGACTGTGGTACCTAGAC

$$h(AC) = 1$$

$$h(AC) = 1 \quad (1 + 1) \% 5 = 2$$

$$h(CT) = 7 \quad (1 + 7) \% 5 = 3$$

$$h(TG) = 14 \quad (1 + 14) \% 5 = 0$$

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

CG .. CT
AC TG
CA AC
CACGACTGTGGTACCTAGAC

Randstrokes

- Pick strobe s_2 given s_1 with equation:

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Example

($n=2$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

CG .. CT
AC TG ...
CA AC GG CT
CAC GACTGTGGT ACCTAGAC
||| ||| ||| ||| ||| |||
CAC -GCTGAGGT -CCTAGAC

Randstrokes

- Pick strobe s_3 given s_1 and s_2 with equation:

$$s_3 = \arg \min_{s' \in W} h(s_1, s_2) + h(s') \mod q$$

Example

($n=3$, $s=2$, $w_{\min}=3$, $w_{\max}=5$)

CA AC GG
CACGACTGTGGTACCTAGAC

Randstrokes

- ✓ Match over substitutions and indels
- ✓ Low ‘redundancy’ between seeds

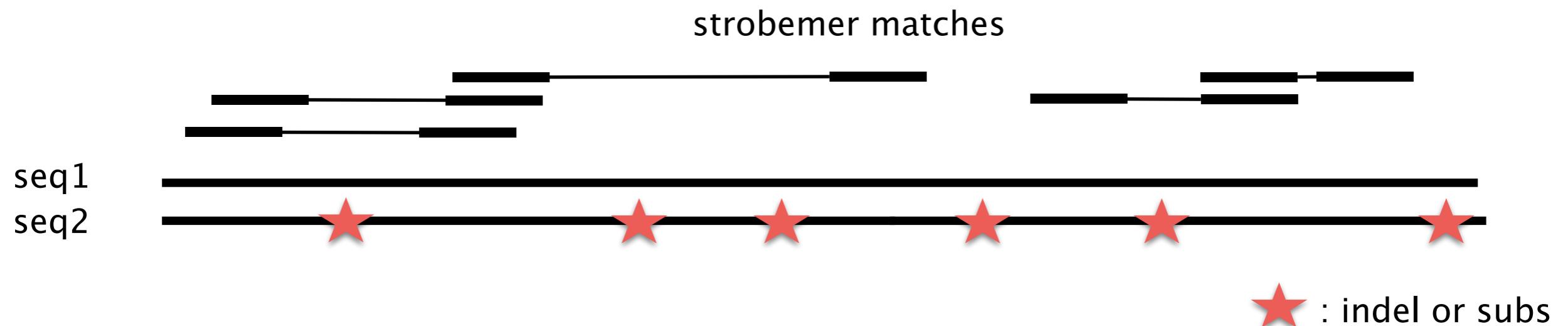
Example

($n=3$, $s=2$, $w_{min}=3$, $w_{max}=5$)

CA AC GG
CACGACTGTGGTACCTAGAC

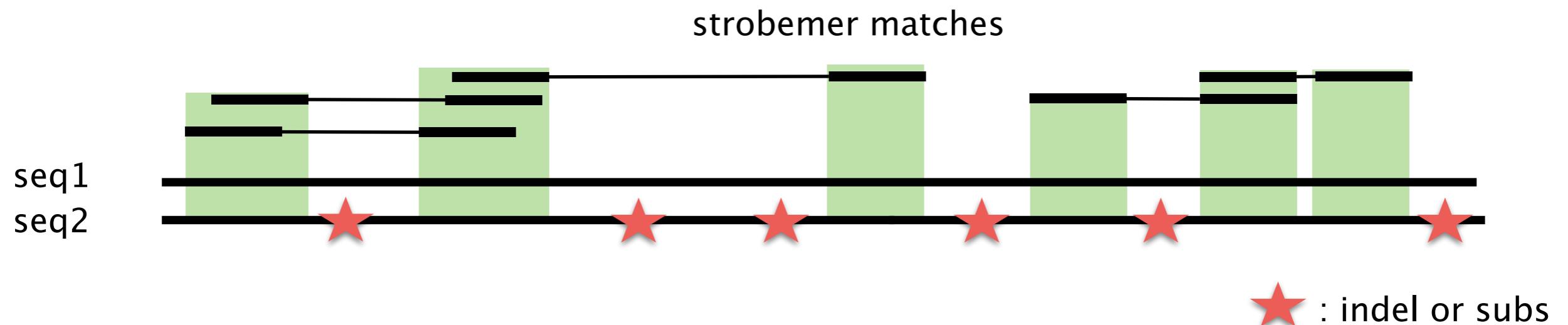
Measuring sensitivity

- Number of matches (m)
 - Sequence coverage (sc)
 - Match coverage (mc)
 - Match gap E-size (E)



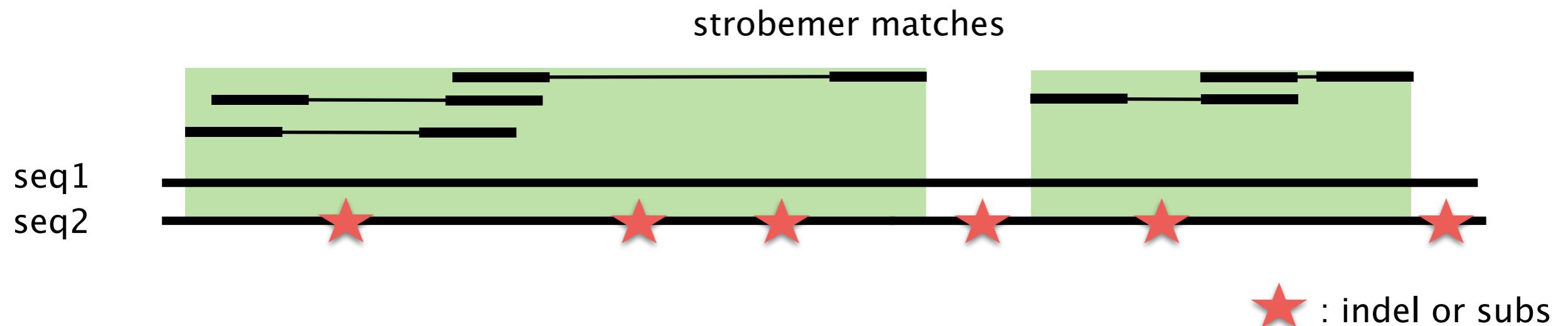
Measuring sensitivity

- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



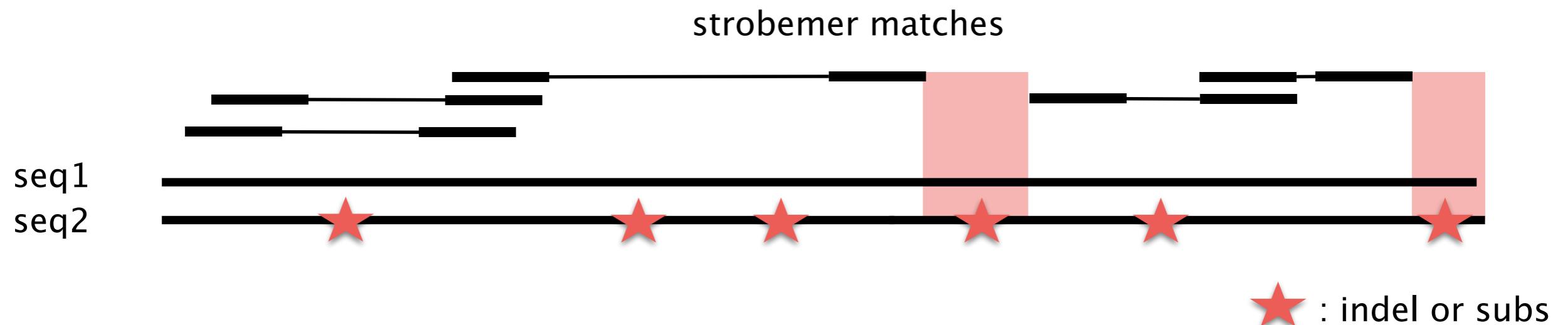
Measuring sensitivity

- Number of matches (m)
- Sequence coverage (sc)
- **Match coverage (mc)**
- Match gap E-size (E)



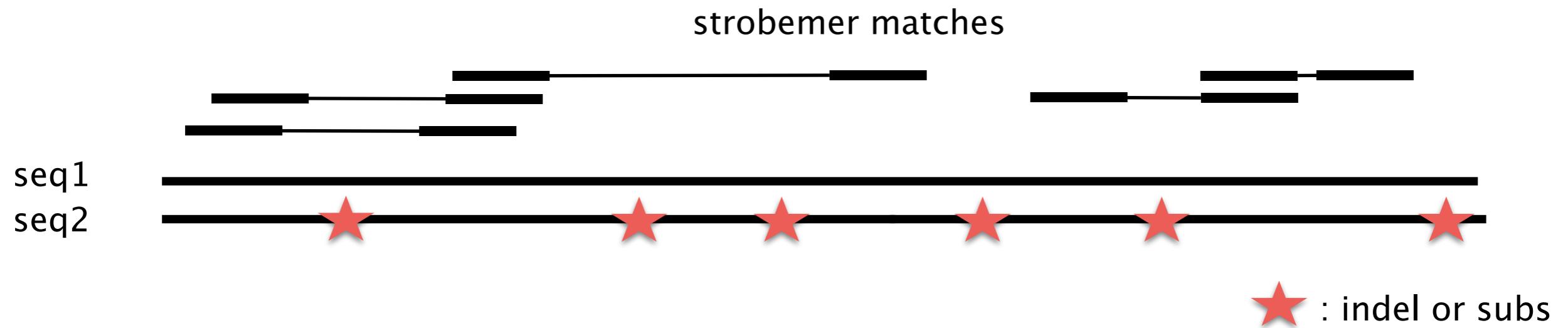
Measuring sensitivity

- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



Randstrokes are sensitive

- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



Construct	length	99%				95%				90%			
		m	sc	mc	E	m	sc	mc	E	m	sc	mc	E
k-mer	30	74.5	95.9	95.9	1.1	22.4	54.7	54.7	43.5	4.7	18.1	18.1	290.2
spaced k-mer	dense	67.6	95.6	96.2	1.5	13.8	50.9	53.9	66.2	1.8	14.1	16.1	491.5
	sparse	50.5	87.8	89.7	11.4	3.5	21.4	26.7	481.4	0.1	2.1	3.6	4053.2
randstrobe	(2,15,25,50)	70.7	98.2	99.9	0.0	18.2	72.7	87.8	8.1	3.4	31.1	44.6	117.7
	(3,10,25,50)	66.7	98.8	100.0	0.0	14.7	78.3	98.2	1.1	2.5	33.7	67.0	68.9

Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
 - Pick a seed uniformly at random from index
 - What is the expected number of hits?

Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
 - Pick a seed uniformly at random from index
 - What is the expected number of hits?

$$E[X] = \dots = \frac{1}{N} \sum_{i=1}^M x_i^2$$

Expected seed count Total seeds Distinct seeds Count seed i

Applicable to any seeding method
A measure of average case query time?

Measuring specificity

1. #Distinct seeds
2. Fraction of them occurring once
3. E-hits (Sahlin, 2022)
 - Pick a seed uniformly at random from index
 - What is the expected number of hits?

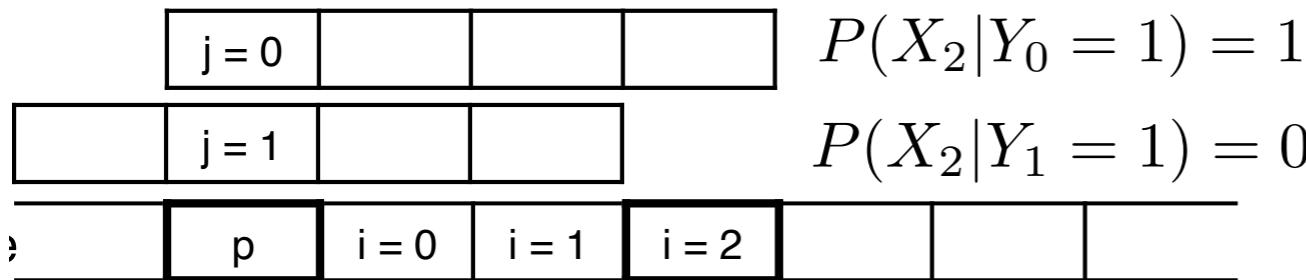
		CHM13		
		% Unique	E-hits	#Distinct (millions)
kmers	48	0.97	152.92	2706.93
randstrobes	(2,24,36,72)	0.98	181.70	2754.53

Conclusion: Comparable specificity for reasonable parameter choices

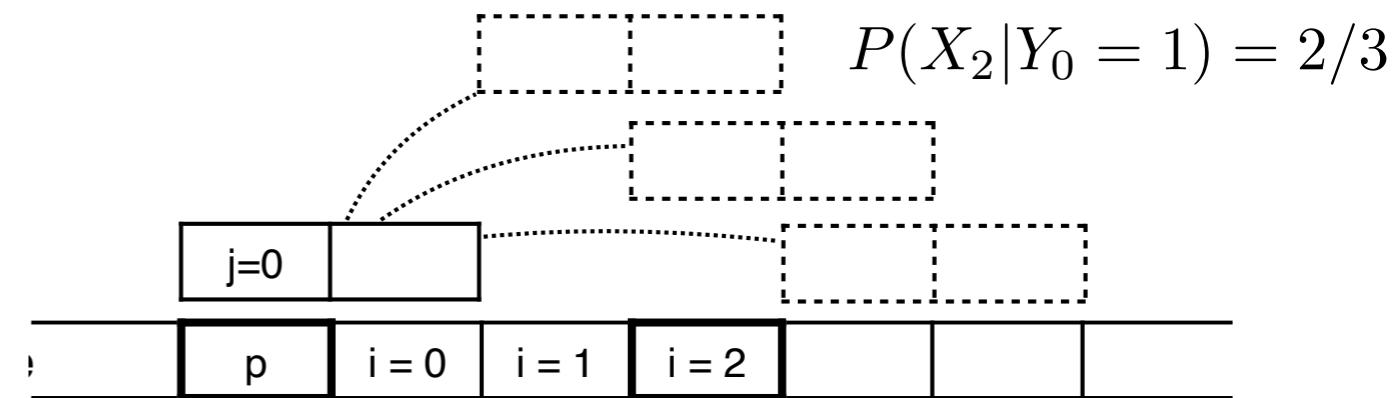
Why are strobemers sensitive?

- Understanding matching sensitivity from a theoretical perspective
- Conditional entropy of seed sampling a position i given that it covers p

k-mers (4)



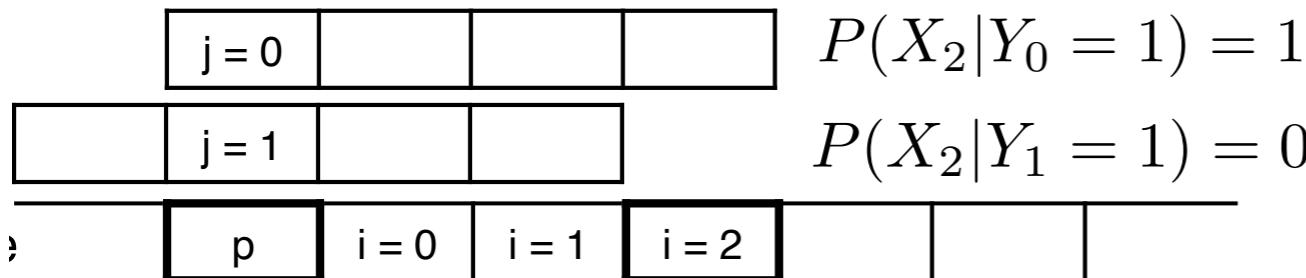
randstrokes (2,2,2,4)



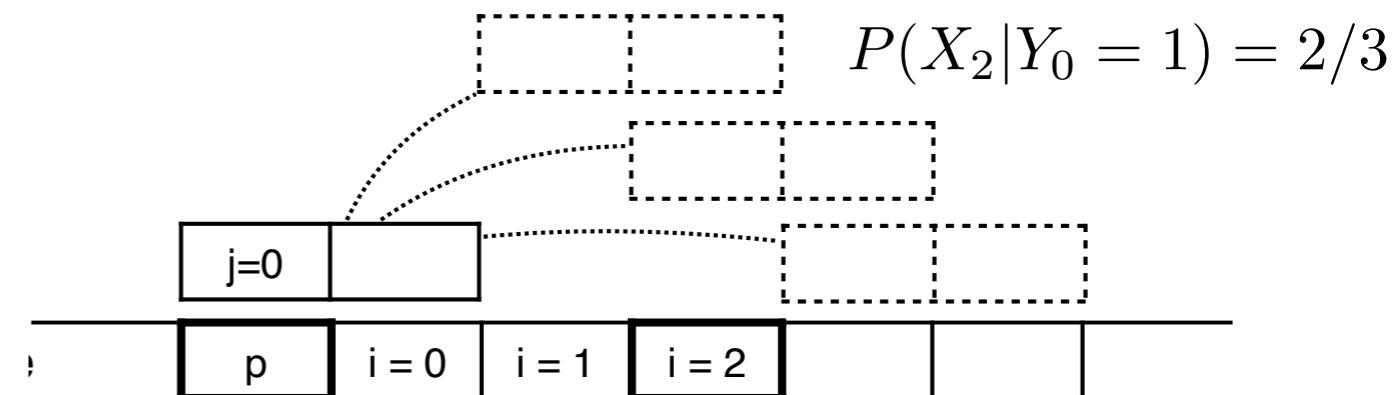
Why are strobemers sensitive?

- Understanding matching sensitivity from a theoretical perspective
- Conditional entropy of seed sampling a position i given that it covers p

k-mers (4)



randstrokes (2,2,2,4)



$$H(X|Y) = - \sum_{j=0}^{k-1} \sum_{i=0}^{w-2} P(Y_j) P(X_i|Y_j) \log_2 P(X_i|Y_j)$$

Can we design seeds with higher entropy?

- Altstrobes: Altering the size of first and second strobe ($s_1 + s_2 = k$)
- Mixedstrobes: Mixing k-mers and randstrobes
- Multistrobes: Selecting strobe size from multiple lengths (still with $s_1 + s_2 = k$)

Randstrobes



Altstrobes



or



Mixedstrobes



or



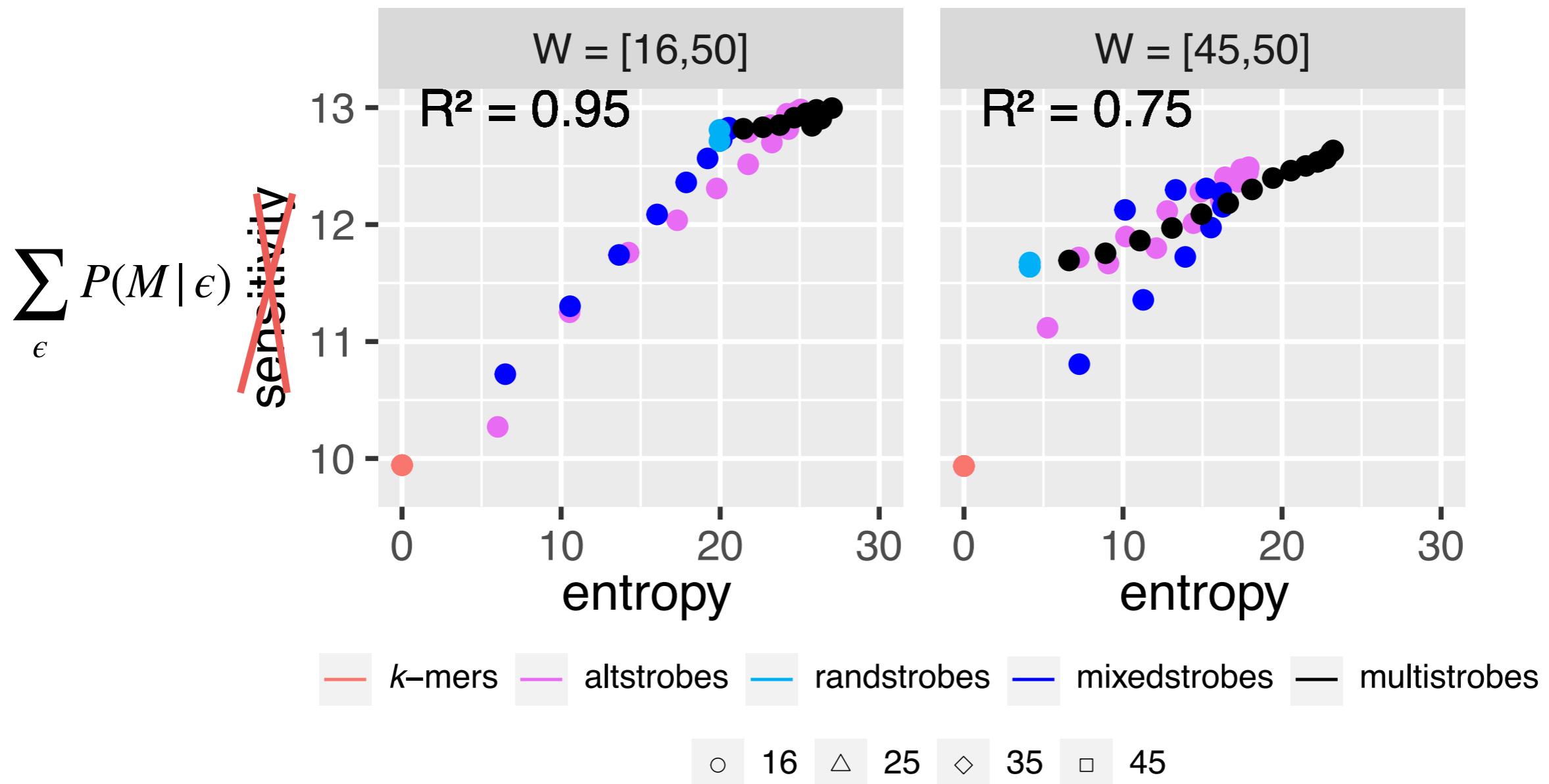
Multistrobes



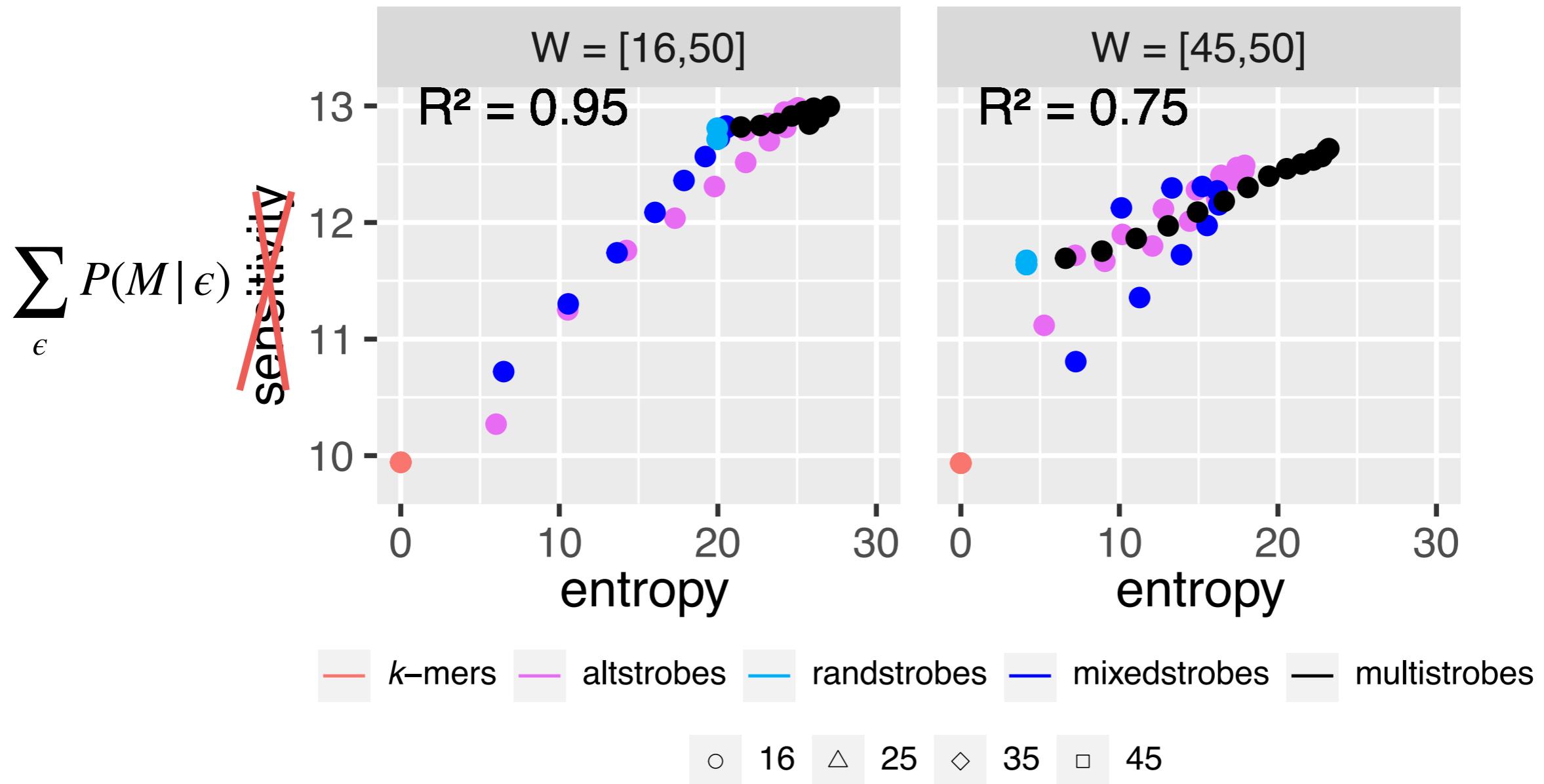
...



Higher entropy leads to higher sensitivity



Higher entropy leads to higher sensitivity



- Result1: Our entropy model predicts seed sensitivity
- Result2: Some seeds produce higher sensitivity than randstrobes

Runtime: Probably not a bottleneck

		Human Chr1
k-mers	30bp	12,4
randstrokes	(2,15,16,40)	19,1
randstrokes	(2,15,16,100)	19,4
randstrokes	(3,10,11,40)	25,1
randstrokes	(3,10,11,100)	27,7

- Result:
 - Number of strobes affects runtime, window size not so much
- Benchmark (in C++):
 - Bitpacking k-mers/strobes
 - **Timings based on Bitwise AND ($h(s_1) + h(s_2)$) & q**

Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \mod q$$

Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \quad \text{and } q$$

Bitwise AND with bitmask (Wei Shen)

Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) \cancel{+} h(s') \text{ m}\cancel{+} d q$$

Bitwise XOR (Lidong Guo ,Giulio Pibiri)

Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \bmod q$$

Variables

- h (e.g., no hash, xxHash, wyhash, Thomas Wang hash – mm2).
- “Link function” – the method to select s_2
- Sampling comparator (e.g., argmin or argmax)
- Final seed hash value $f(s_1, s_2)$

Can we optimise the implementation?

$$s_2 = \arg \min_{s' \in W} h(s_1) + h(s') \bmod q$$

Variables

- h (e.g., no hash, xxHash, wyhash, Thomas Wang hash – mm2).
- “Link function” – the method to select s_2
- Sampling comparator (e.g., argmin or argmax)
- Final seed hash value $f(s_1, s_2)$

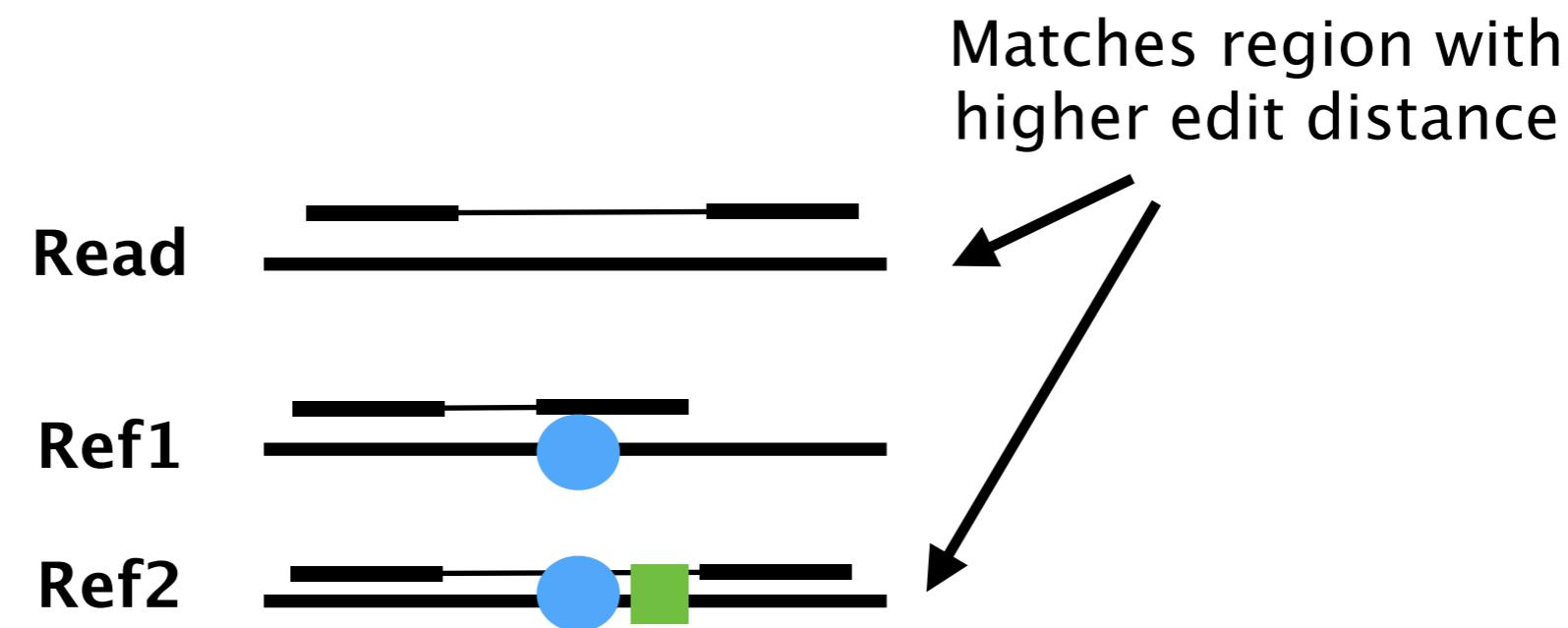
Objectives

- Fast
- Maximize entropy of sampled strobe in window

Results

- Link method influences both time and randomness

Limitation: Approximate seeds do not guarantee matching the most similar region 😞



Implications?

- May not be reliable for applications that only count matches
- Using positions (for chaining/clustering) can alleviate the issue

Strobemers are useful in practice

Other groups

- RNAblloom¹: Reference-free long-read transcriptome assembly
- BLEND²: Long-read overlap detection
- Symbiont-screener³: Reference-free long read phasing based on “markers”
- SAKE⁴: Identify all k-mers in a genome from long reads

Our work

- strobealign⁵: Short-read alignment.

¹Nip et al, Nat Commun, 2023

²Firtina et al., NAR genom. bioinform., 2023

³Xu et al., Front. mar. sci., 2023.

⁴Leinonen and Salmela, Plos One, 2023.

⁵Sahlin, Genome Biology, 2022.

Short-read mapping with strobealign

- strobealign¹ combines syncmers² and randstrokes³
- Main idea: Can afford to use longer seeds

```
GGTCGATTGGACTCTGCCGGAGTTGCATCTTGGCTCAGCAGACCGGCCTCCGTCGCCGTTATAACCGACCCAGGGGGGGGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGGAGAGC...
 TTGGACTCTGCCGGAGTTGC
  GCCGGAGTTGCATCTTGG
   GAGTTGCATCTTGGCTCA
    GCATCTTGGCTCAGCAGA
     TTTGGCTCAGCAGACCGGC
      TCAGCAGACCGGCCTCCG
       AGACCGGCCTCCGTC
        CTCCCGTCGTGCCGTTAT
         GTCGTGCCGTTATAACGC
          CGCCGTTATAACCGACCC
           ATAACCGACCCAGGGGGGG
            ACCCAGGGGGGGGAGTTCC
             GGGGGGGAGTTCCGGATGC
              CGAGTTCCGGATGCCATCCA
               CGGGATGCCATCCACGATGC
```

Short-read mapping with strobealign

- strobealign¹ combines syncmers² and randstrokes³
- Main idea: Can afford to use longer seeds

Strobe from window of syncmers instead of k-mers

¹Sahlin, Genome Biology, 2022.

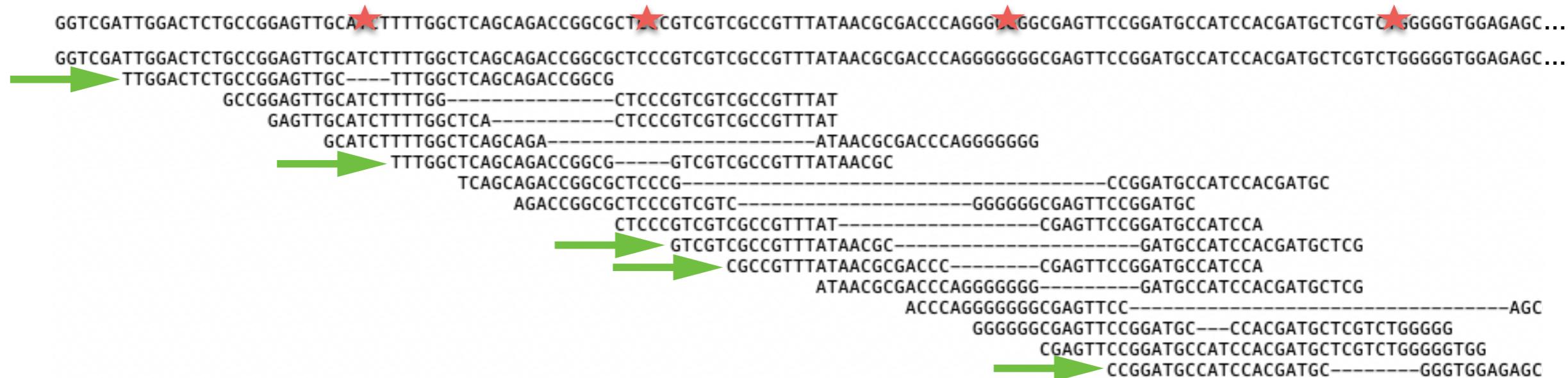
² Edgar, PeerJ, 2020

³ Sahlin, Genome Research. 2021

Short-read mapping with strobealign

- strobealign¹ combines syncmers² and randstrokes³
- Main idea: Can afford to use longer seeds

Reference



GGTCGATTGGACTCTGCCGGAGTTGCA★TTTTGGCTCAGCAGACCGGGCGT★CGTCGTGCCGTTATAACCGCACCCAGGG★GGCGAGTTCCGGATGCCATCCACGATGCTCGT★GGGGTGGAGAGC...

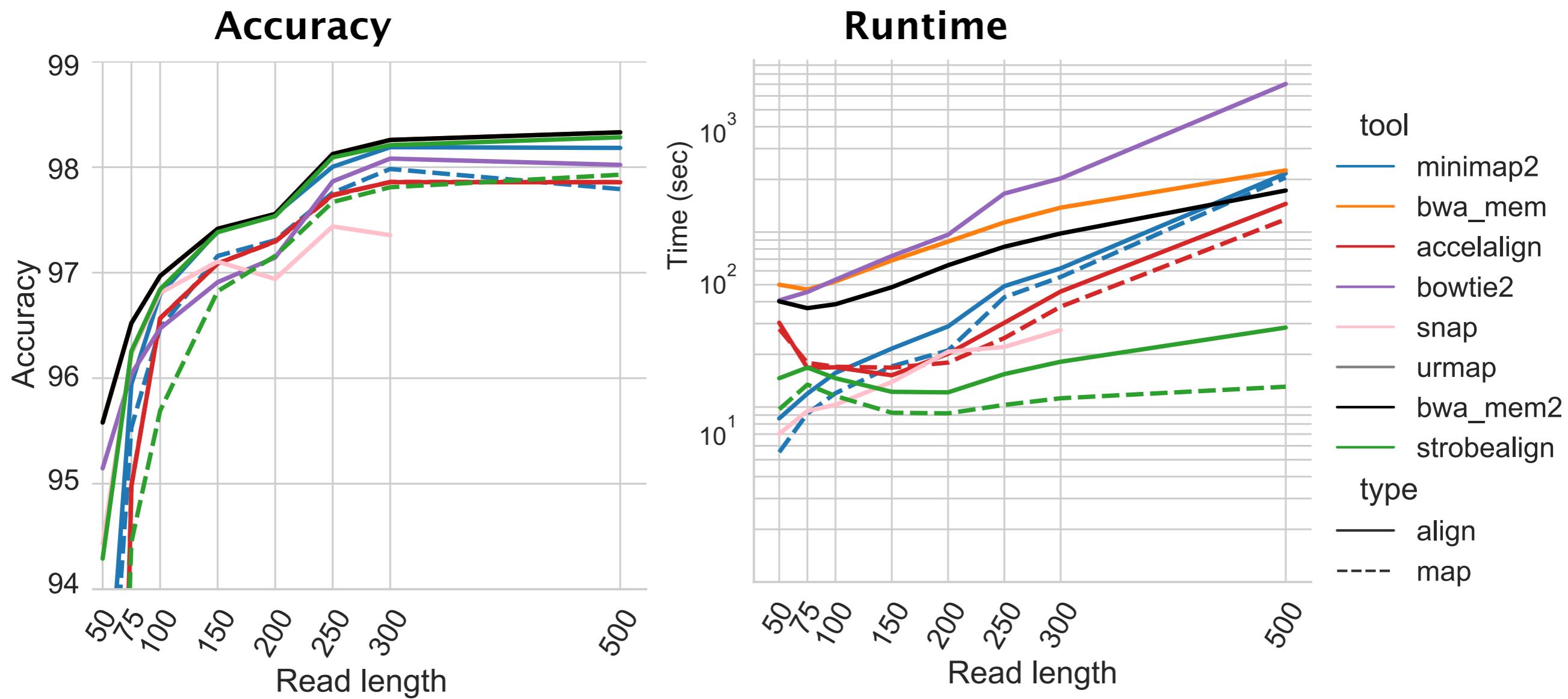
GGTCGATTGGACTCTGCCGGAGTTGCATCTTGCTCAGCAGACCGGGCGTCCCGTCGTGCCGTTATAACCGCACCCAGGGGGCGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGGAGAGC...

→ TTGGACTCTGCCGGAGTTGC---TTTGGCTCAGCAGACCGGGCG
GCCGGAGTTGCATCTTG-----CTCCCCTCGTCGCCGTTTAT
GAGTTGCATCTTGCTCA-----CTCCCCTCGTCGCCGTTTAT
GCATCTTGCTCAGCAGA-----ATAACCGCACCCAGGGGGGG
→ TTTGGCTCAGCAGACCGGGCG---GTCGTGCCGTTATAACGC
TCAGCAGACCGGGCGCTCCCG-----CCGGATGCCATCCACGATGC
AGACCGGGCGCTCCCGTC-----GGGGGGCGAGTTCCGGATGC
CTCCCCTCGTCGCCGTTTAT-----CGAGTTCCGGATGCCATCCA
→ GTCGTGCCGTTATAACGC-----GATGCCATCCACGATGCTCG
CGCCGTTATAACCGCACCC-----CGAGTTCCGGATGCCATCCA
ATAACCGCACCCAGGGGGGG-----GATGCCATCCACGATGCTCG
ACCCAGGGGGGGCGAGTTCC-----AGC
GGGGGGCGAGTTCCGGATGC---CCACGATGCTCGTCTGGGGGG
CGAGTTCCGGATGCCATCCACGATGCTCGTCTGGGGTGG
→ CGGGATGCCATCCACGATGC-----GGGTGGAGAGC

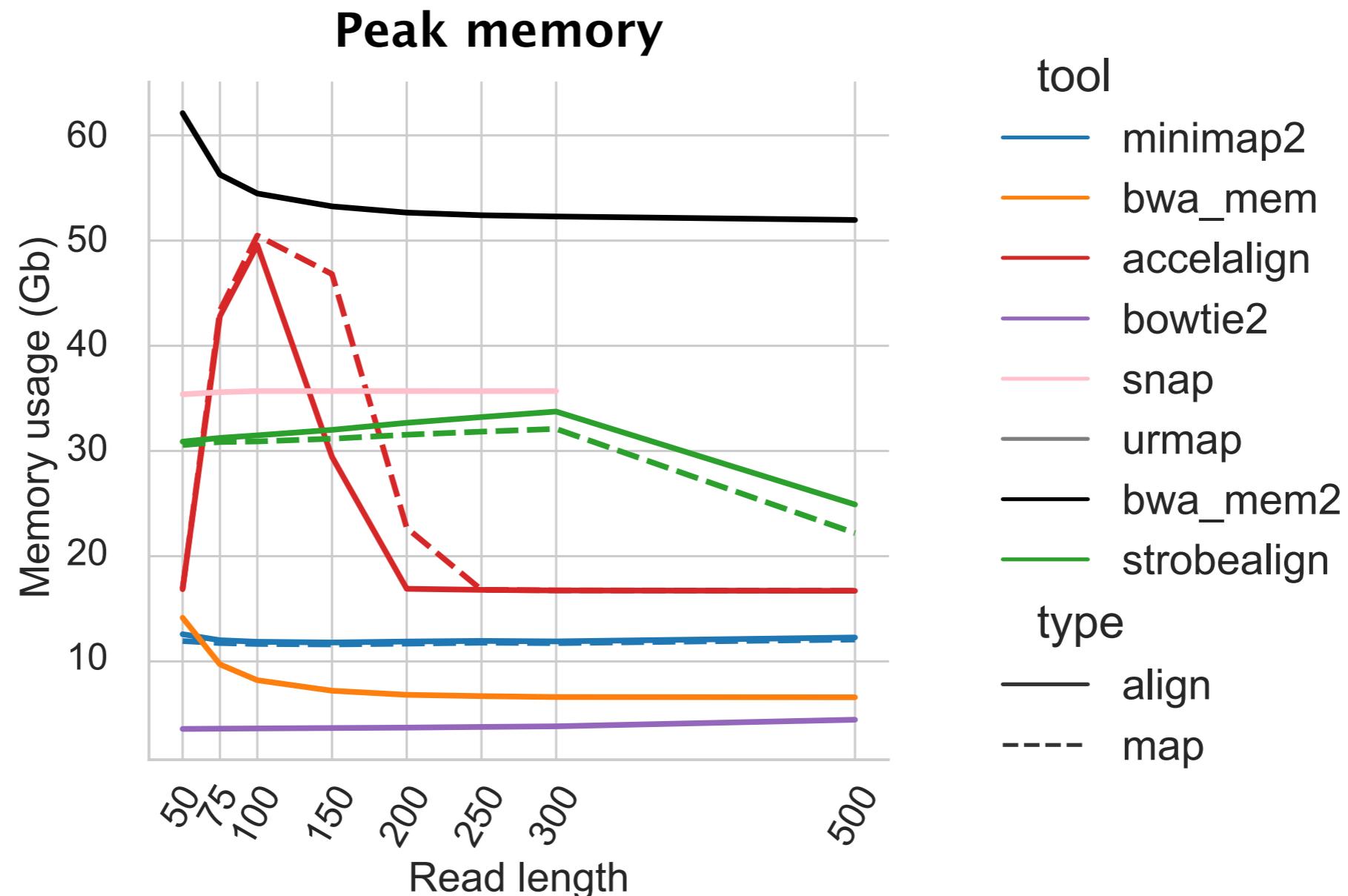
Chance to match

Strobealign has high accuracy ($\geq 150bp$) and fast

- Paired-end reads at different variation rates to hg38
- Tools run with 16 threads



But high memory consumption!

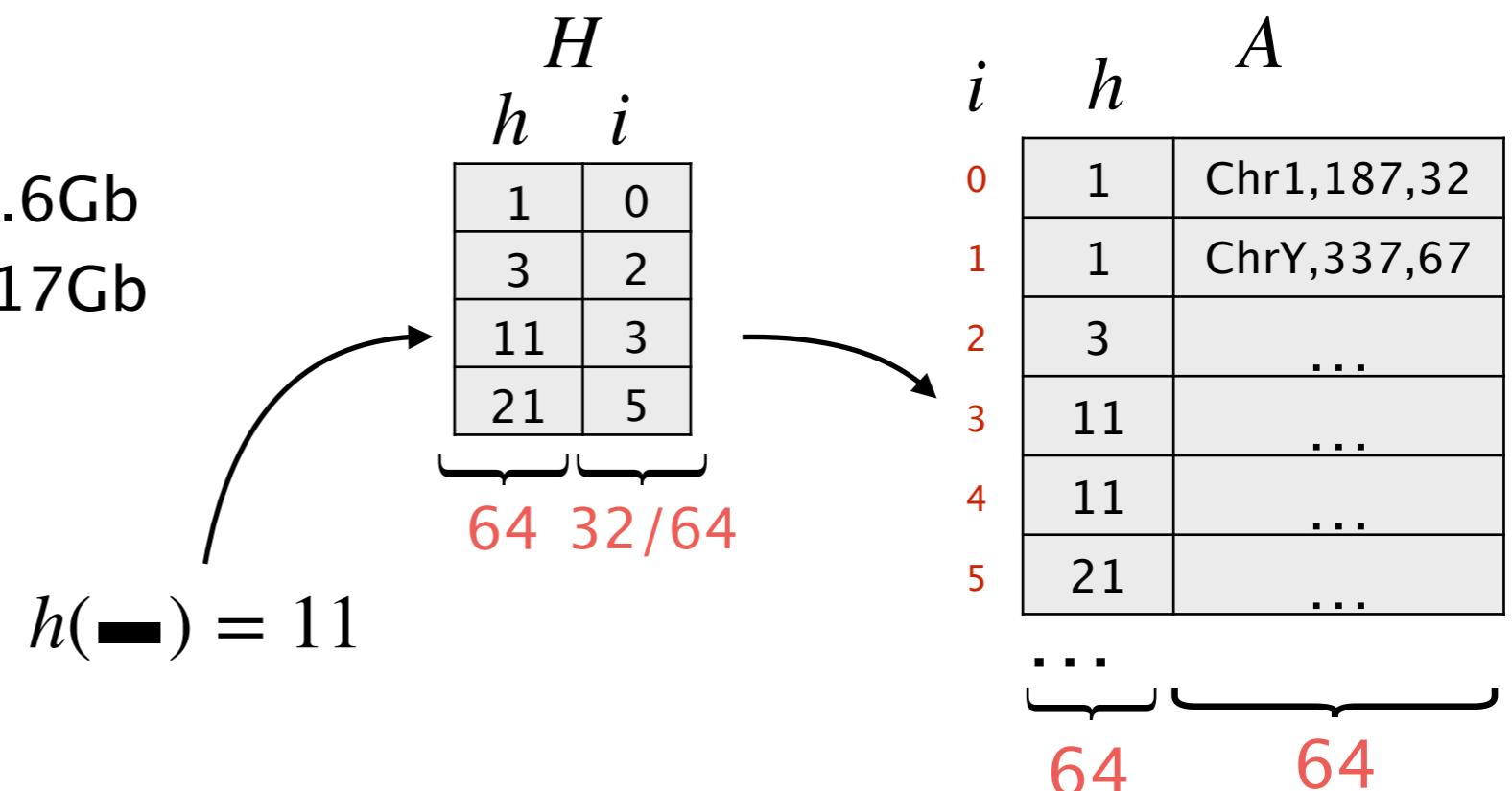


What is consuming memory?



Human

- Genome: 3Gb
- $A: 3B*(1/5)*128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$

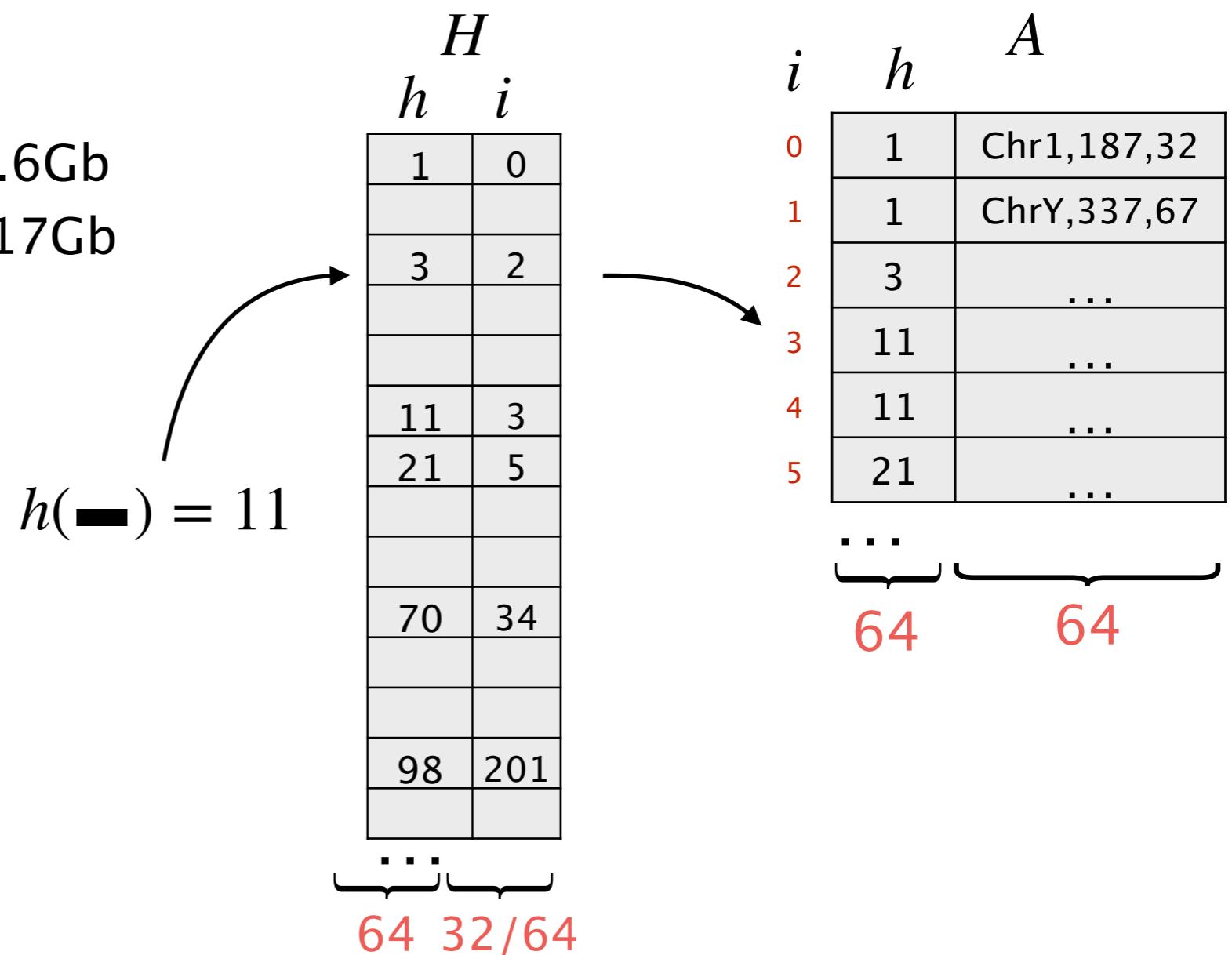


What is consuming memory?



Human

- Genome: 3Gb
- $A: 3B * (1/5) * 128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$

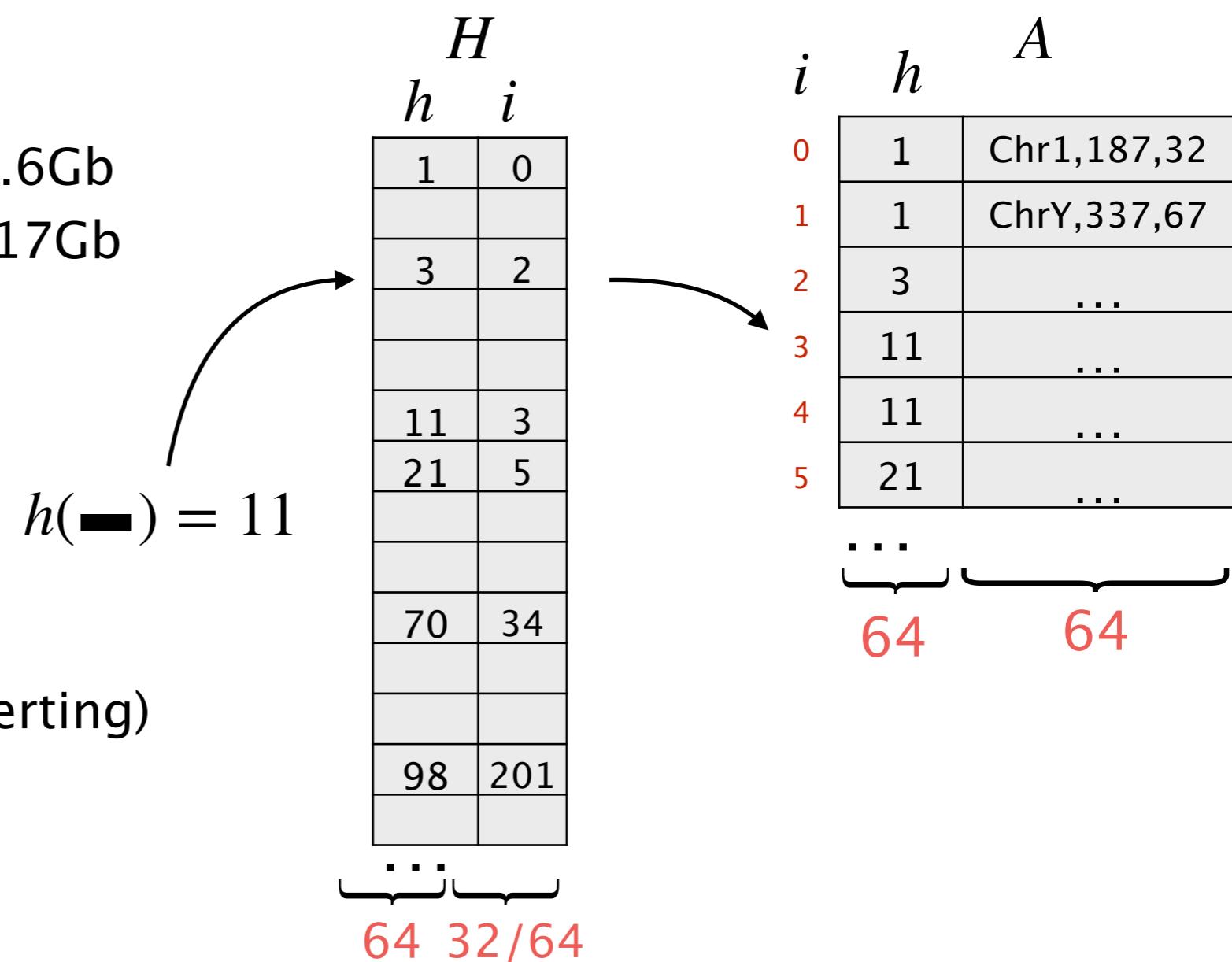


What is consuming memory?



Human

- Genome: 3Gb
- $A: 3B * (1/5) * 128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$



Idea (Cred. Luis Coelho)

- A is static (no deleting/inserting)
- Hash table not needed
- Use prefix lookup vector!
- $N = 28 \rightarrow 268\text{Mb}$

What is consuming memory?

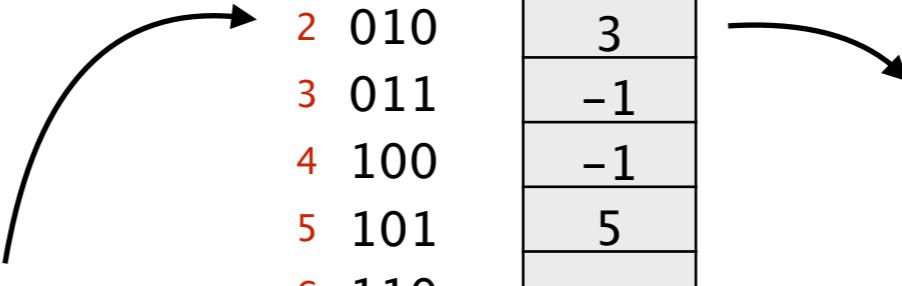
Reference



Human

- Genome: 3Gb
- $A: 3B * (1/5) * 128 \text{ bits} = 9.6\text{Gb}$
- $H = \text{load factor etc...?} \approx 17\text{Gb}$

$$h(\text{---}) = 11 = \boxed{01011}$$

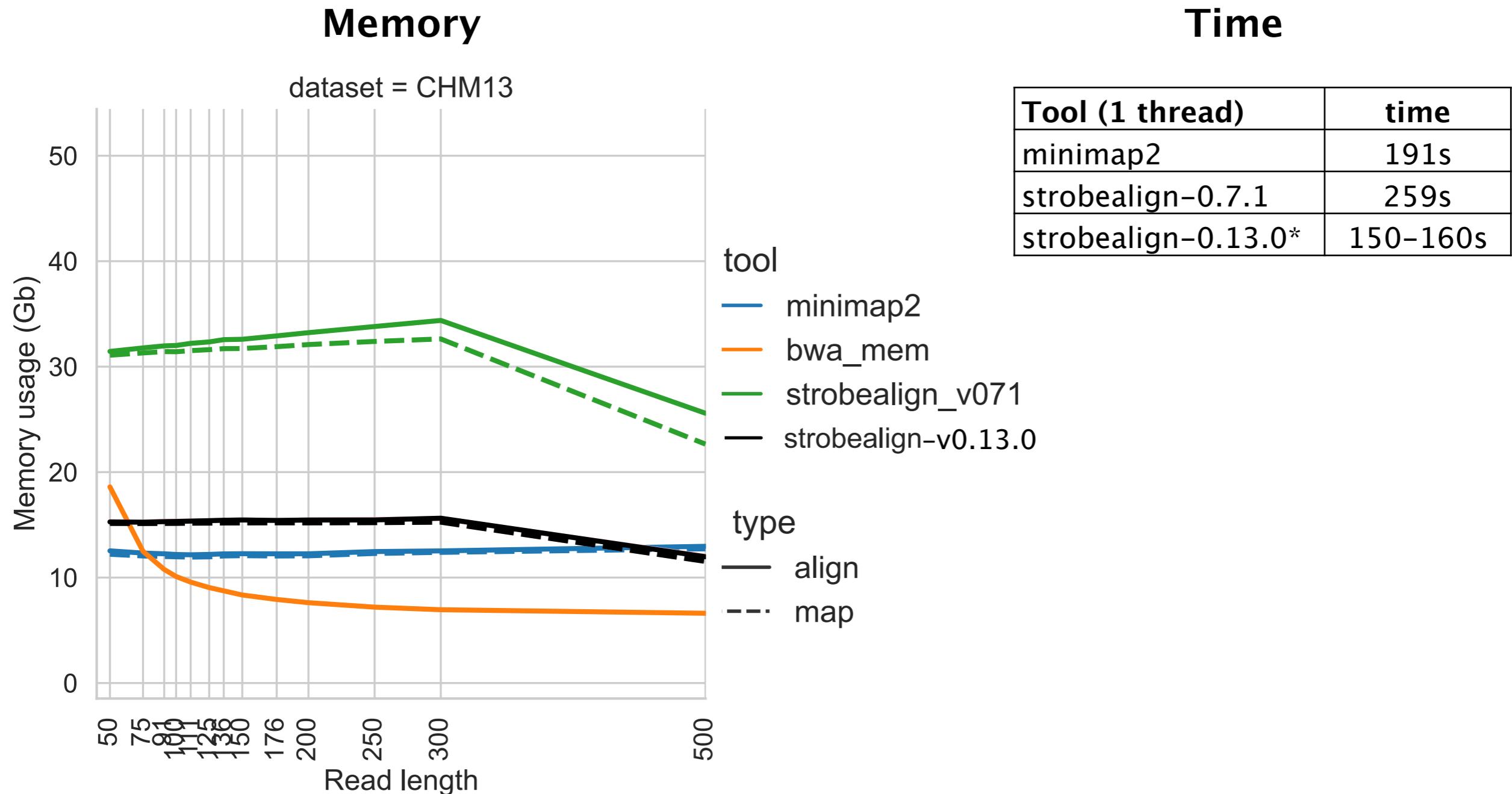


	H	A
	$N = 3$ (2^N)	i
0	000	00001 Chr1,187,32
1	001	00001 ChrY,337,67
2	010	00011 ...
3	011	01011 ...
4	100	01011 ...
5	101	10101 ...
6	110	...
7	111	...
		...
		64
		64

Idea (Cred. Luis Coelho)

- A is static (no deleting/inserting)
- Hash table not needed
- Use prefix lookup vector!
- $N = 28 \rightarrow 268\text{M slots} \rightarrow \sim 2\text{Gb}$

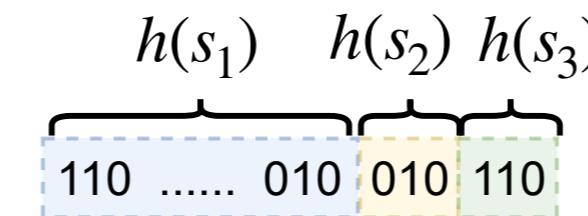
A prefix-lookup table reduces index size (and lowers construction time)



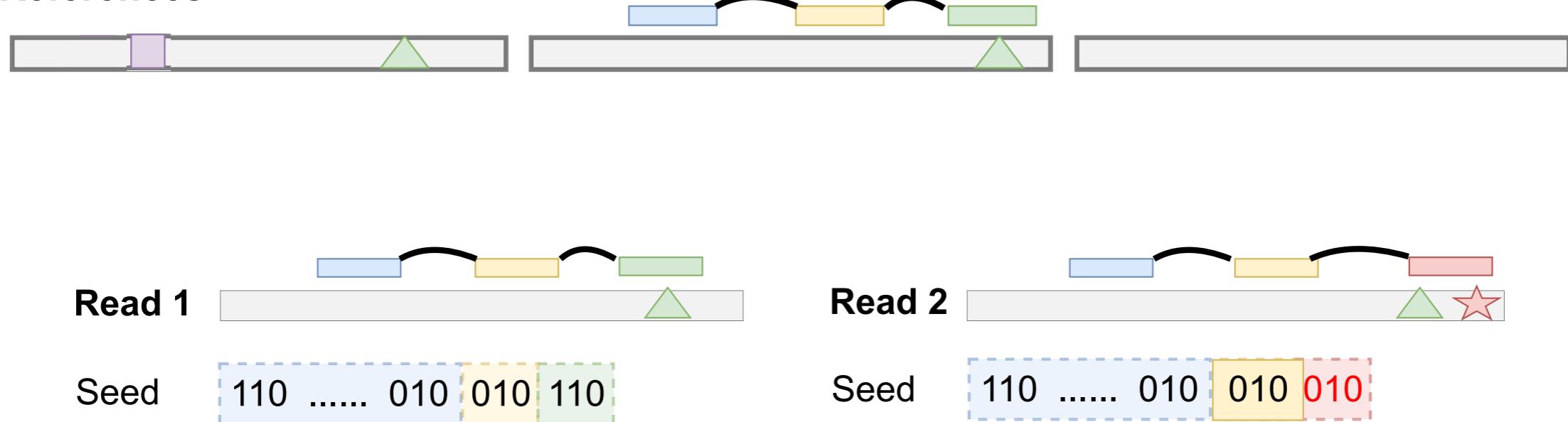
Multi-context seeds (MCS) improves accuracy

Strobe information lost!

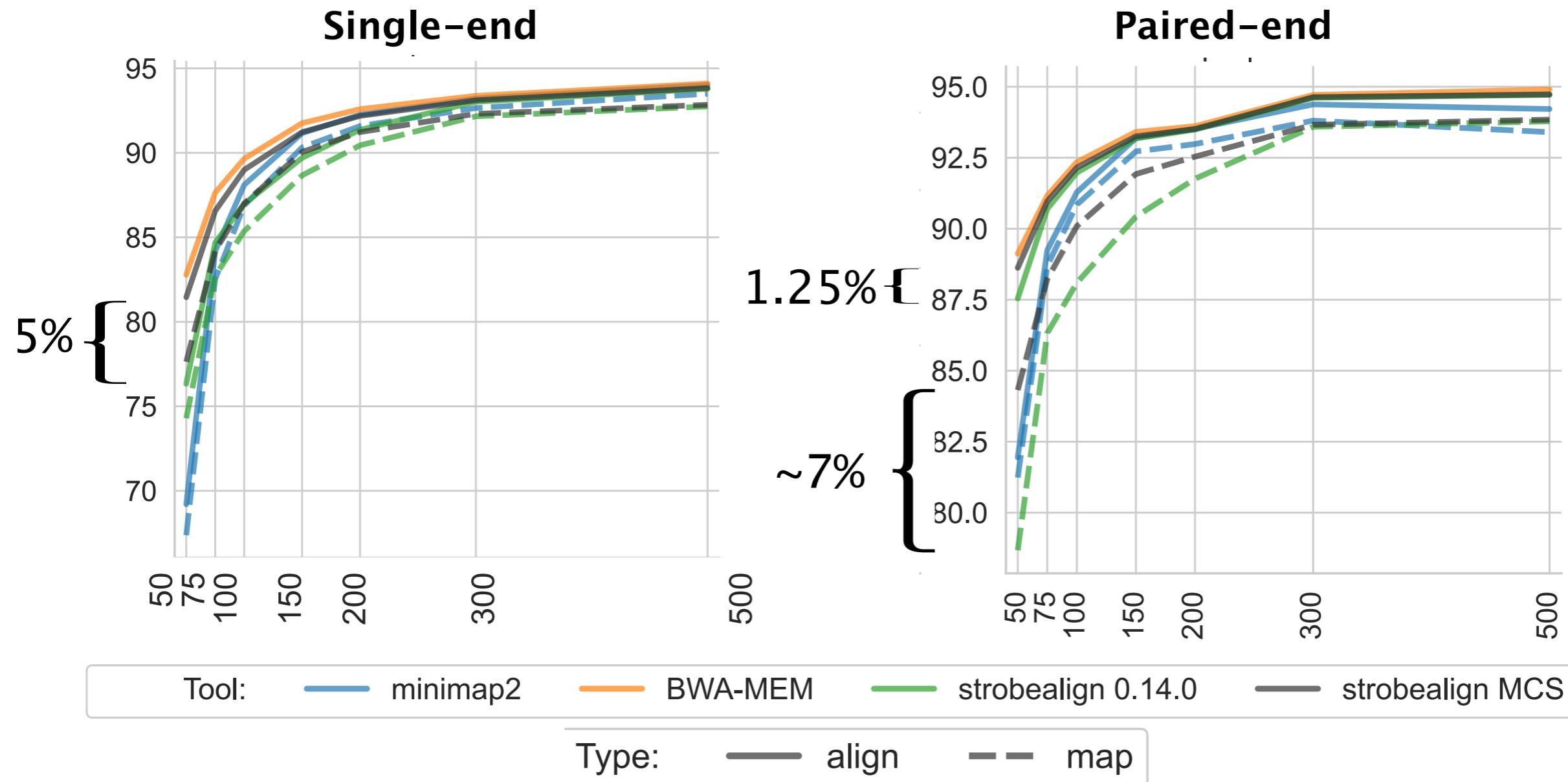
- Strobemers seed hash: $f(s_1, s_2)$, something like $2h(s_1) - h(s_2)$ or $h(s_1) + h(s_2)$
- Idea: store hashes of strobes individually



References

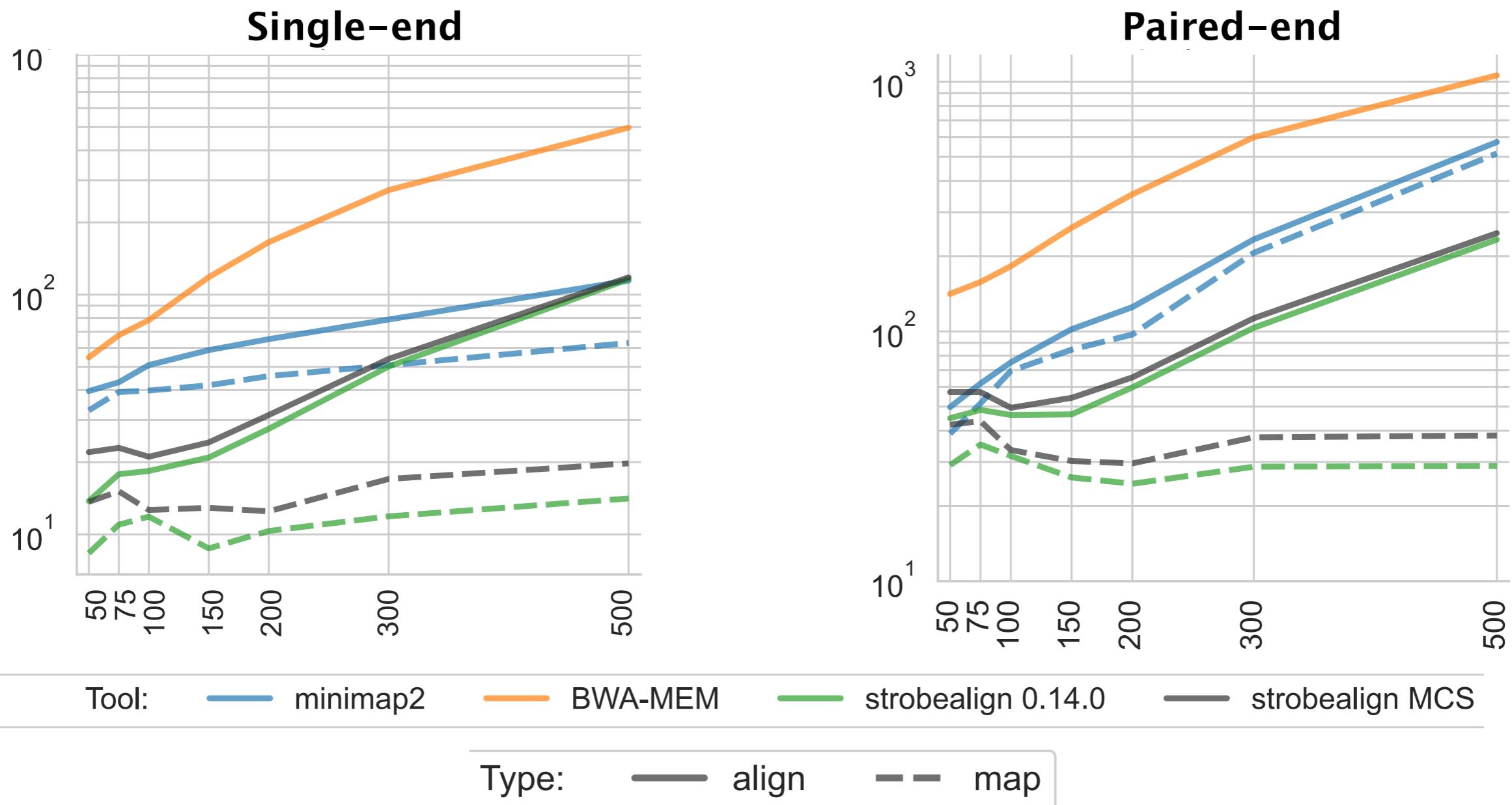


Mapping accuracy (simulated Illumina reads to CHM13)



Mapping time (simulated Illumina reads to CHM13)

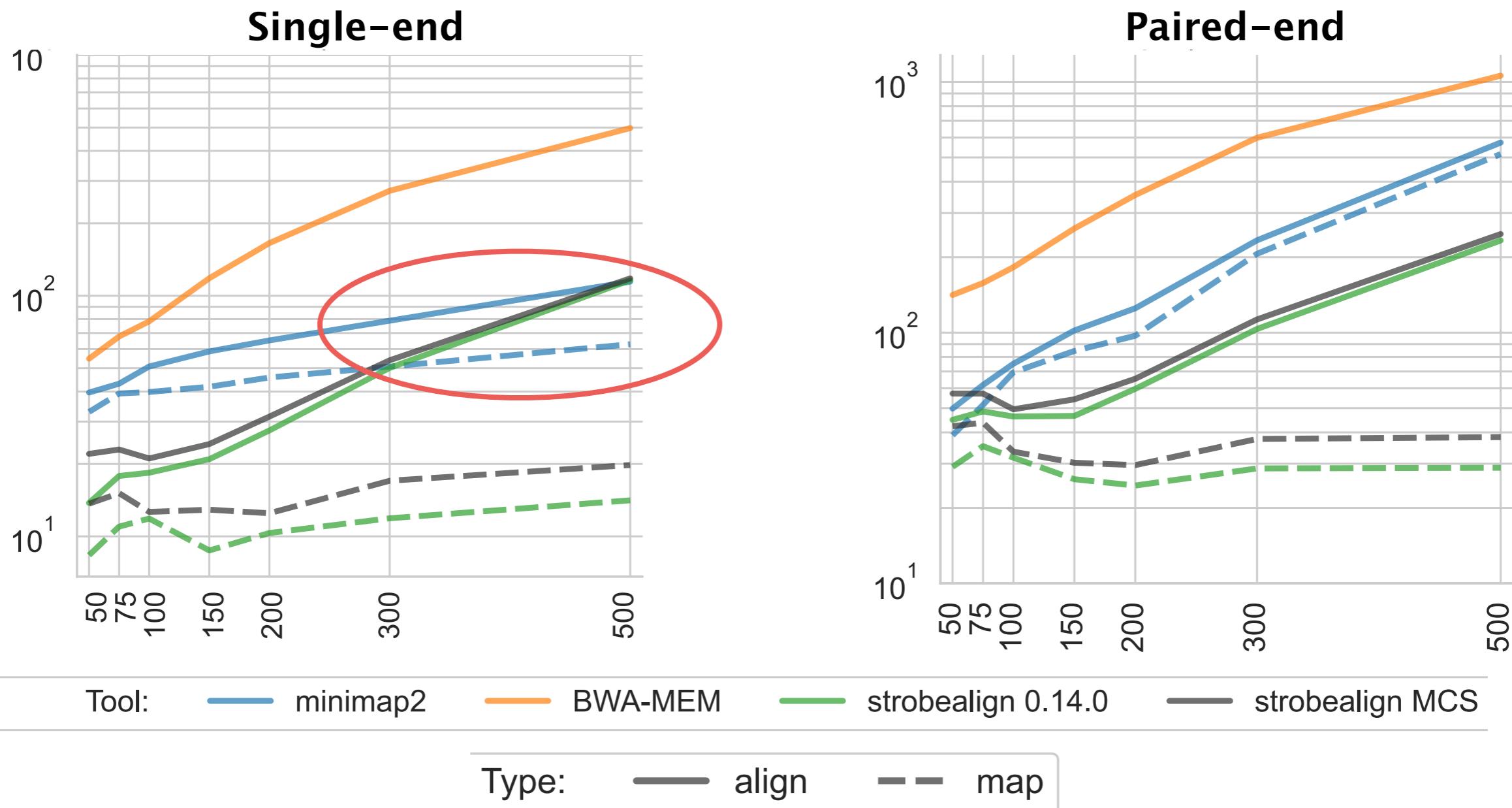
- strobealign-MCS is a bit slower than v0.14.0 (more seeds)
- We believe we can optimise aemb a bit



*Tolstoganov et al., 2024, bioRxiv

Mapping time (simulated Illumina reads to CHM13)

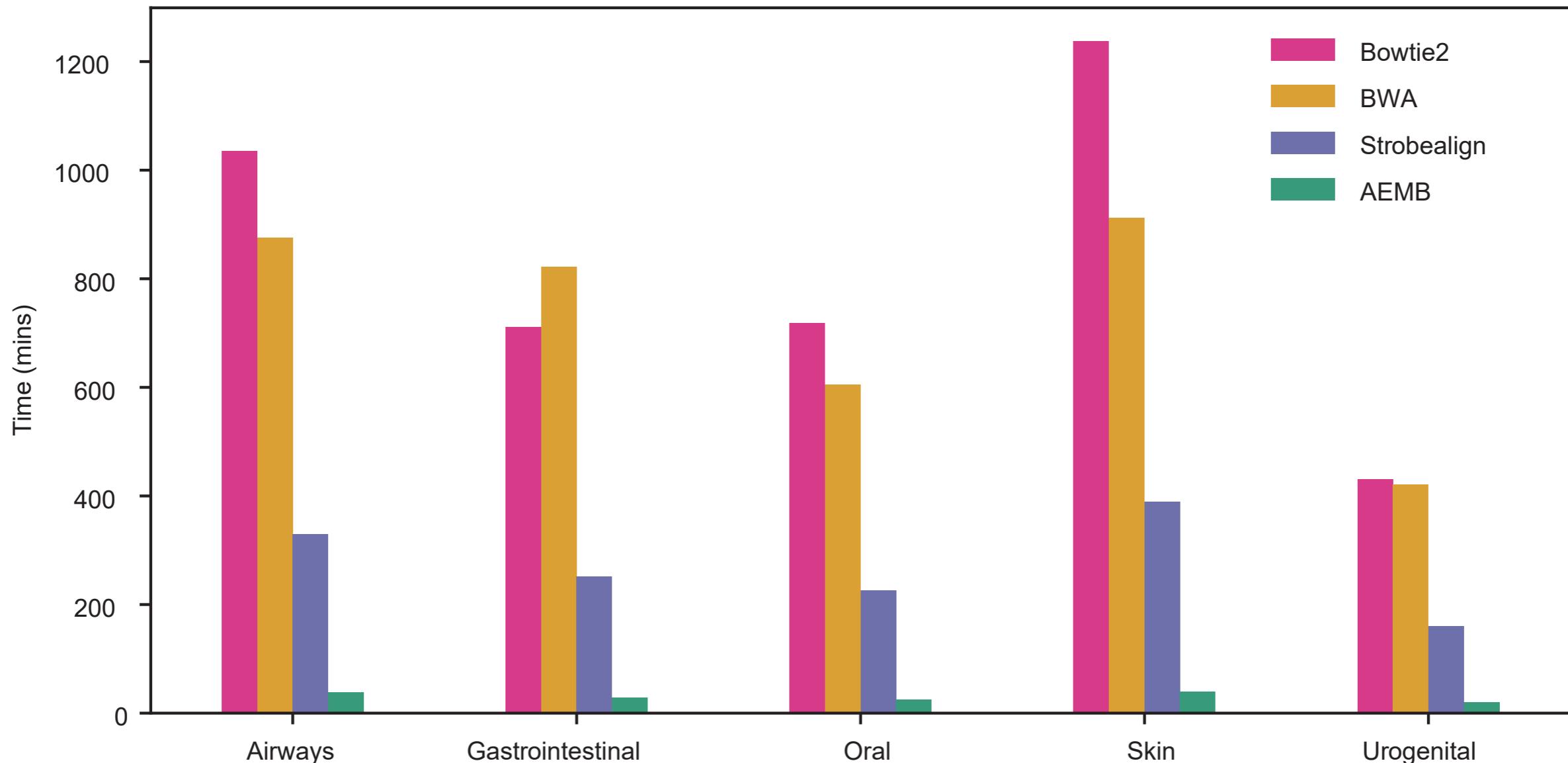
- strobealign-MCS is a bit slower than v0.14.0 (more seeds)
- We believe we can optimise aemb a bit
- **Another conclusion: We need to do piece-wise (extension is a bottleneck)**



strobealign new feature: --aemb

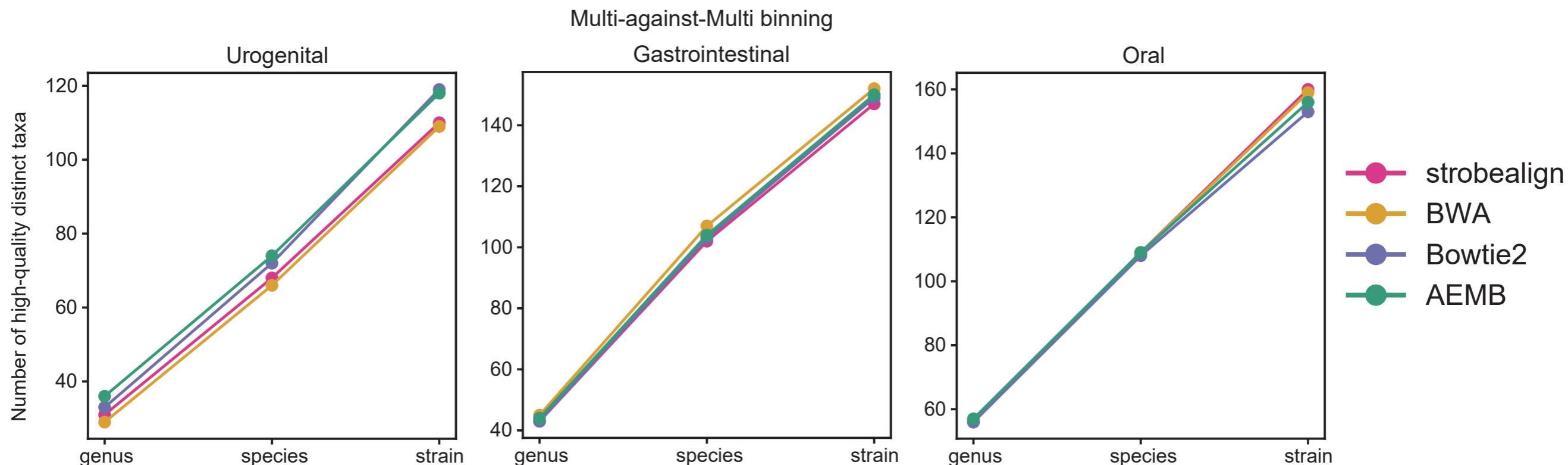
- **aemb**: “Abundance estimation metagenomic (contig) binning”
- Mapping-based coverage computation
- Multi-against-multi binning (Mattock and Watson, 2023)

4 threads



strobealign new feature: --aemb

- **aemb**: “Abundance estimation metagenomic (contig) binning”
- Mapping-based coverage computation
- Multi-against-multi binning (Mattock and Watson, 2023)



Conclusions and future work

Strobemers

- Sensitive and specific (Sahlin, 2021)
- E-hits: a measure for seed repetitiveness given a reference (Sahlin, 2022)
- Entropy ‘randomness’ predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

Conclusions and future work

Strobemers

- Sensitive and specific (Sahlin, 2021)
- E-hits: a measure for seed repetitiveness given a reference (Sahlin, 2022)
- Entropy ‘randomness’ predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

Strobealign

- Fast and accurate – for some lengths.. (Sahlin, 2022)
- Lower memory with prefix-table (Pan et al., forthcoming)
- Accurate for all lengths with MCS (Tolstoganov et al., 2024)
- aemb: rapid metagenomic coverage computation (Pan et al., forthcoming)

Conclusions and future work

Strobemers

- Sensitive and specific (Sahlin, 2021)
- E-hits: a measure for seed repetitiveness given a reference (Sahlin, 2022)
- Entropy ‘randomness’ predicts sensitivity (Maier and Sahlin, 2023)
- Implementation affects the practical result (Karami et al., 2024)
- Multi-context seeds (MCS): query full and partial seed (Tolstoganov et al., 2024)

Strobealign

- Fast and accurate – for some lengths.. (Sahlin, 2022)
- Lower memory with prefix-table (Pan et al., forthcoming)
- Accurate for all lengths with MCS (Tolstoganov et al., 2024)
- aemb: rapid metagenomic coverage computation (Pan et al., forthcoming)

Future work

- Split-read mapping: enabled with MCS
- Long-read mapping
- External-memory version
- (Piecewise extension, altstrobes)

Acknowledgements

Strobemers

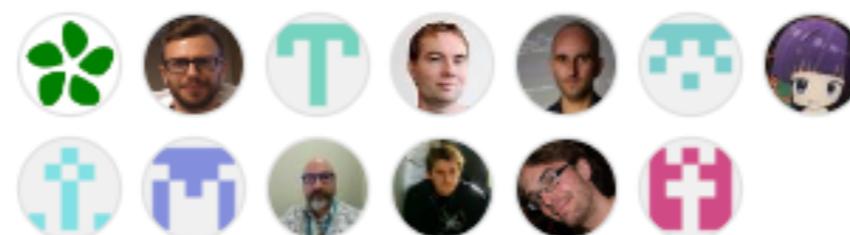
Benjamin Maier
Moein Karami
Aryan Soltani
Wei Shen
Rob Patro
Giulio Pibri
Lidong Guo
Barış Ekim
Mengyang Xu
Daniel Liu
Heng Li

Strobealign

Marcel Martin
Ivan Tolstoganov
Luis Coelho
Shaojun Pan
Johan Gustavsson
Sebastian Jaenicke
Tom Conway
Paul Theodor Pyl
Martin Larralde
Richèl Bilderbeek

...

Contributors 13



 @ksahlin

 @ksahlin.bsky.social

 @krsahlin

Funding

SciLifeLab

 Vetenskapsrådet
Swedish Research Council


NATIONAL BIOINFORMATICS
INFRASTRUCTURE SWEDEN





Funded by
the European Union

Appendix

E-hits

x_i : count seed i

$p_i = x_i/N$

N : total seeds, M : distinct seeds

$$E[X] = \sum_{i=1}^M x_i p_i = \sum_{i=1}^M x_i \frac{x_i}{N} = \frac{1}{N} \sum_{i=1}^M x_i^2$$

Conclusions Moein et al?

- Link method influences both time and randomness
- Link method $\text{XOR } (h(s_1) \oplus h(s_1))$ varies with argmin vs argmax

XOR with argmin: selects $s_2 = s_1$ if present

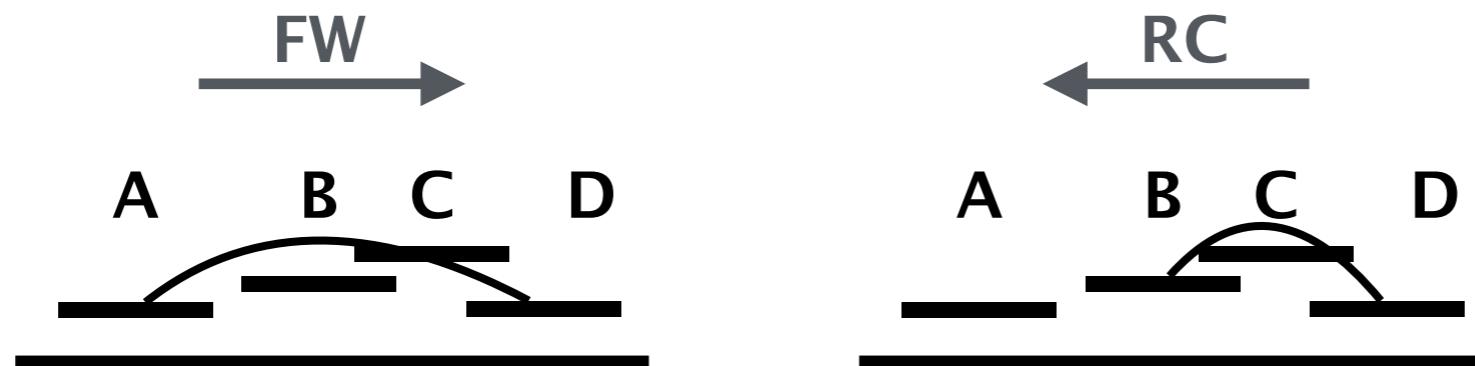


XOR with argmax – variations are selected for



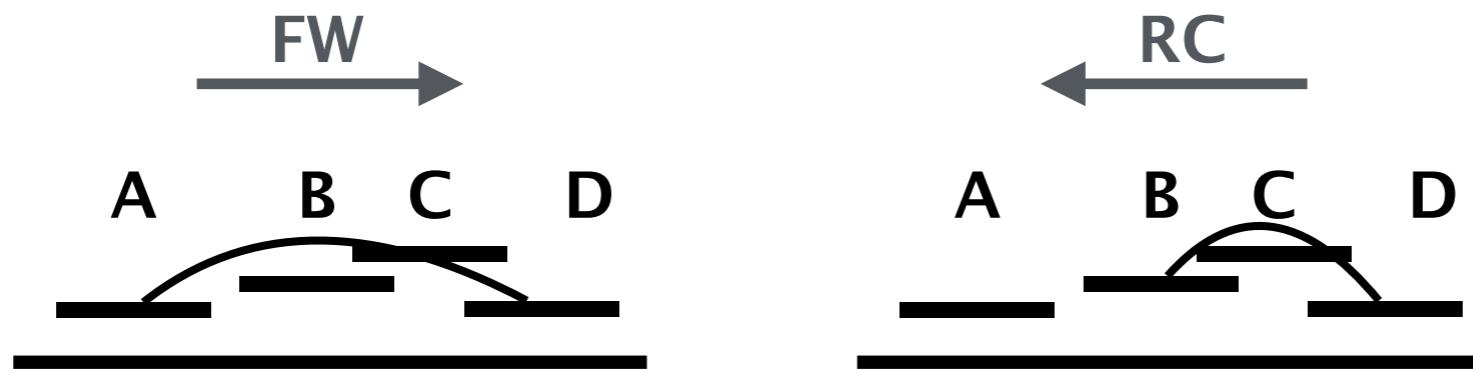
Strobemers: Limitations

1. No guarantee to match most similar region 😞
2. True canonicity requires twice the work 😞



Strobemers: Limitations

1. No guarantee to match most similar region 😞
2. True canonicity requires twice the work 😕



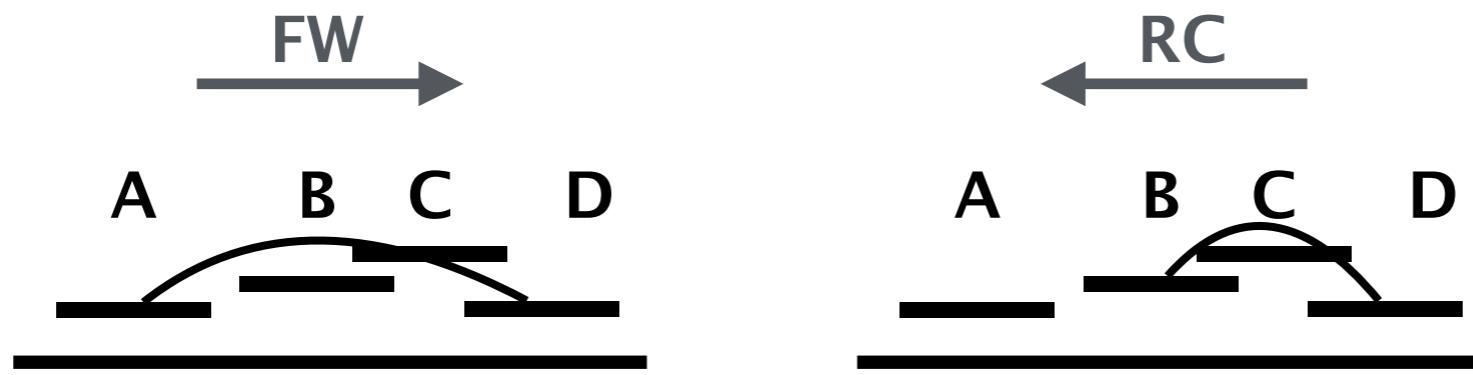
Remember: $f(X, Y)$ computes the hash value of the seed

$$\min\{f(A, D), f(D, B)\}$$

Full canonicity

Strobemers: Limitations

1. No guarantee to match most similar region 😞
2. True canonicity requires twice the work 😕



Remember: $f(X, Y)$ computes the hash value of the seed

$$\min\{f(A, D), f(D, B)\}$$

Full canonicity

Symm. $f(A, D) = f(D, A)$

Ex: A + B

'Some canonicity' (lol)

Asymm. $f(A, D) \neq f(D, A)$

Ex: 2A - B

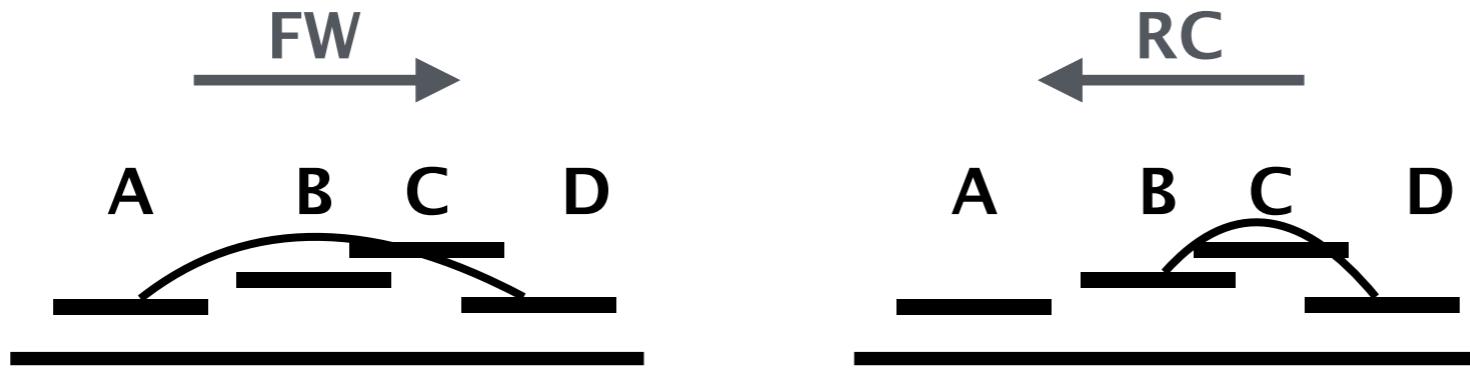
No canonicity

Strobemers: Limitations

1. No guarantee to match most similar region



2. True canonicity requires twice the work



Remember: $f(X, Y)$ computes the hash value of the seed

$$\min\{f(A, D), f(D, B)\}$$

Full canonicity

Symm. $f(A, D) = f(D, A)$

Ex: $A + B$

'Some canonicity' 😊

Asymm. $f(A, D) \neq f(D, A)$

Ex: $2A - B$

No canonicity

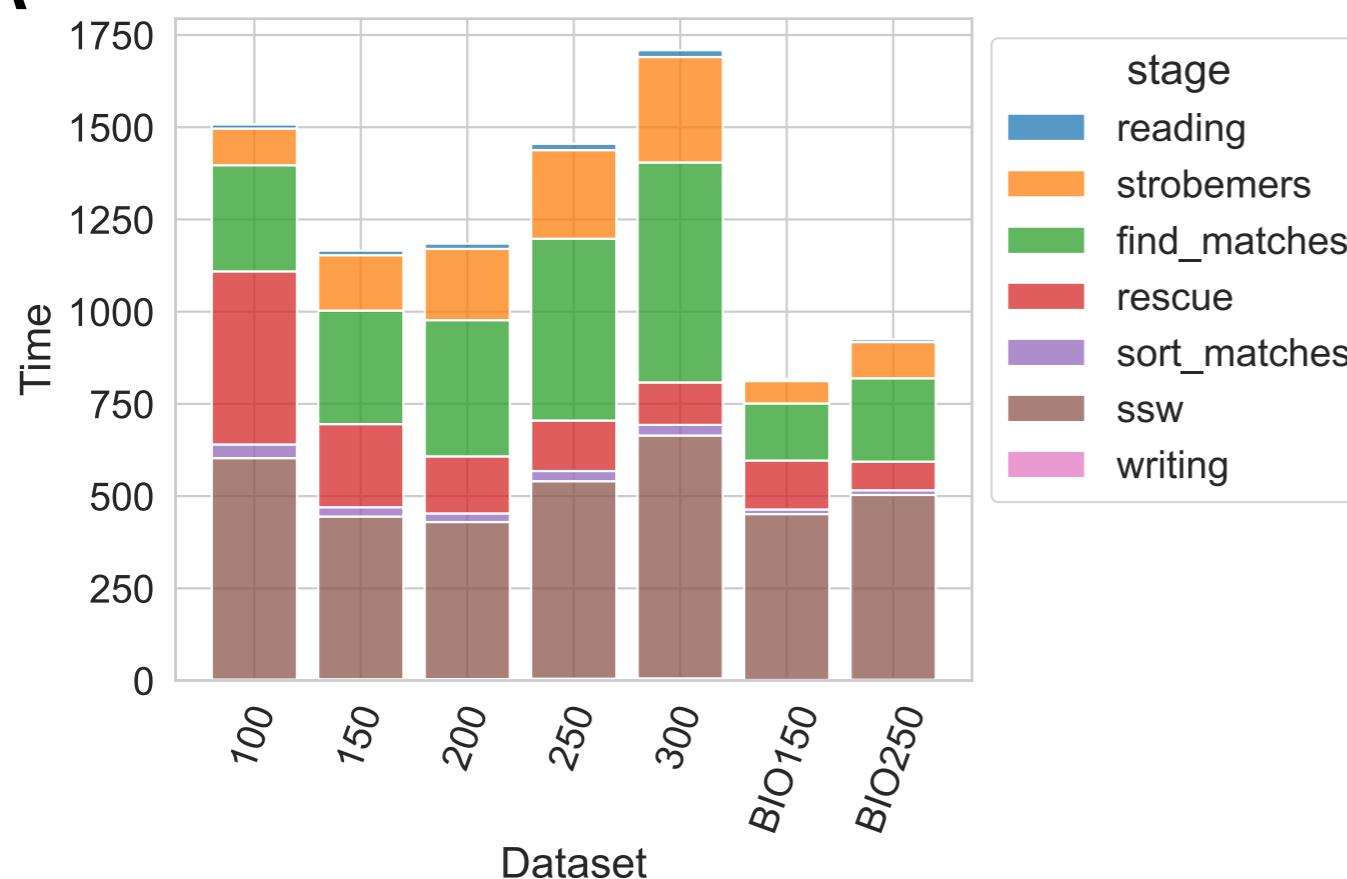
Indexing: one pass (FW)

Querying: both FW and RC

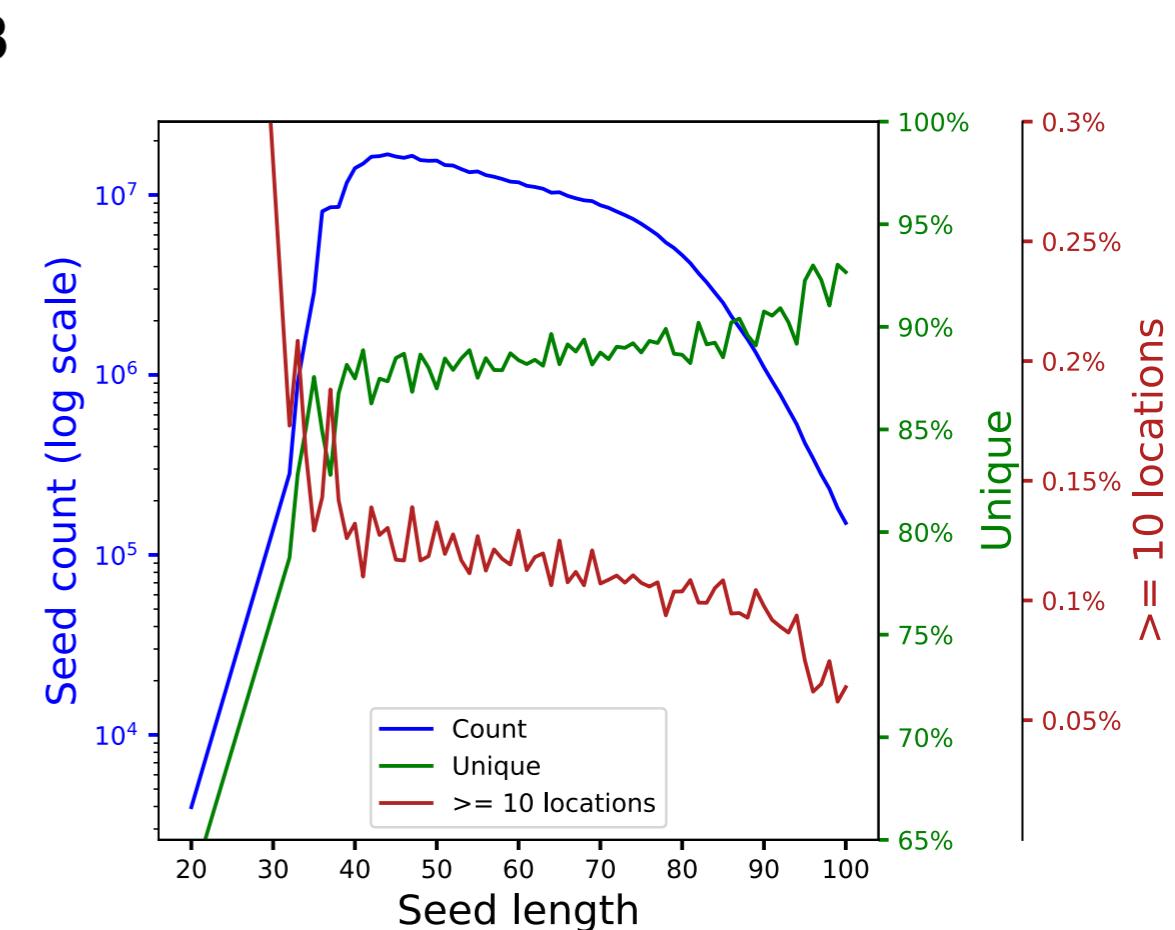
Symmetric f speeds up indexing, but at some cost:
inversions together with masking repeat masking seeds

Time allocation

A



B



E-hits: Expected seed hits

- draw a seed uniformly at random from index
 - What is the expected number of hits?

x_i : count seed

$$p_j = x_j/N$$

N : total seeds, M : distinct seeds

$$E[X] = \sum_{i=1}^M x_i p_i = \sum_{i=1}^M x_i \frac{x_i}{N} = \frac{1}{N} \sum_{i=1}^M x_i^2$$

