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)

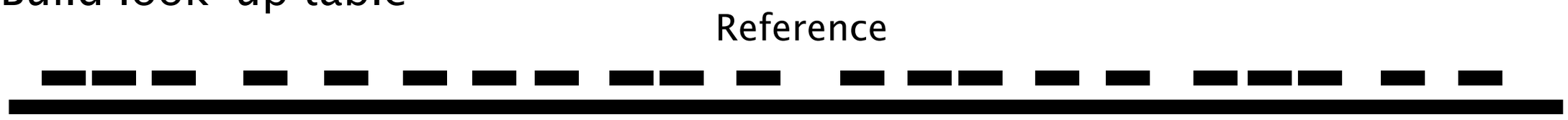
Read mapping

- 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

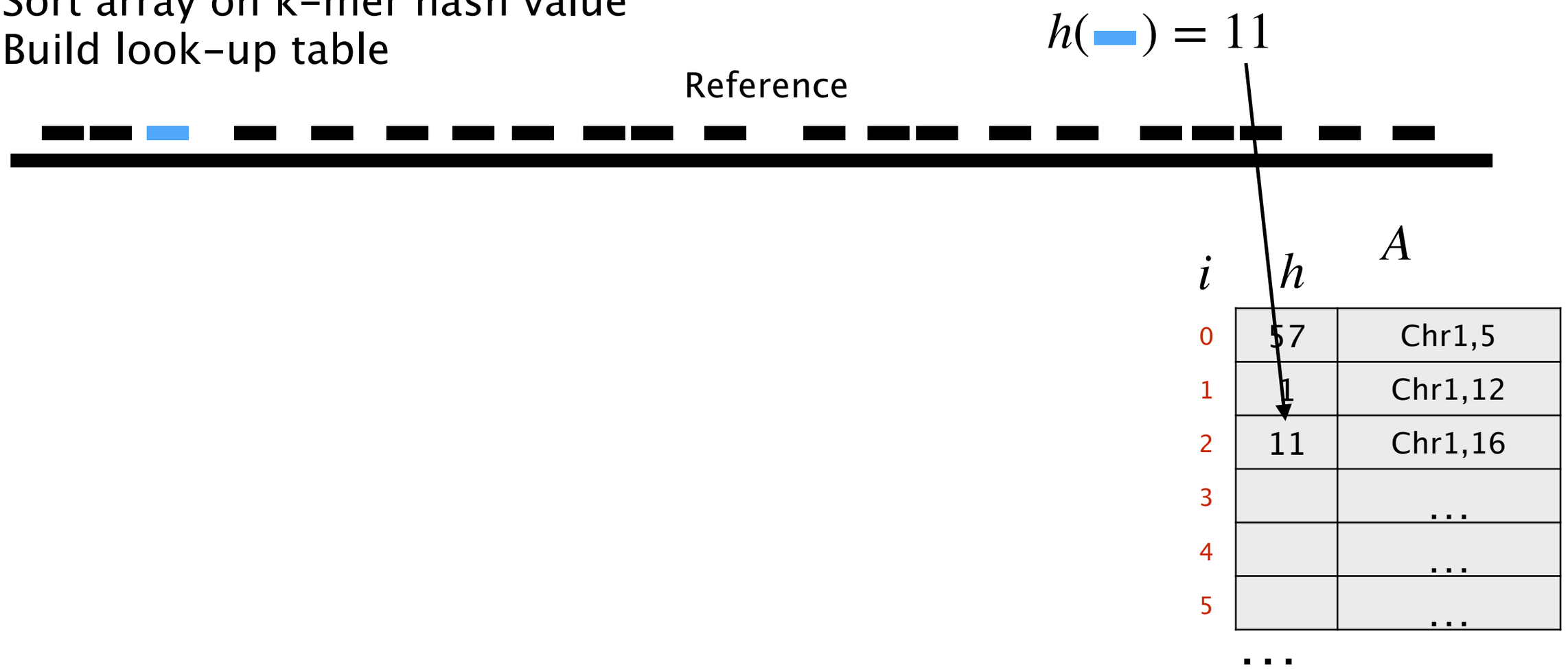
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



Hash-based seed-filter-extend

Indexing

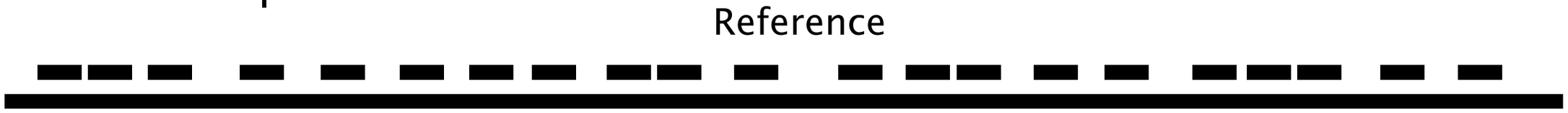
1. Extract k-mers, hash, and put in array
2. Sort array on k-mer hash value
3. Build look-up table



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

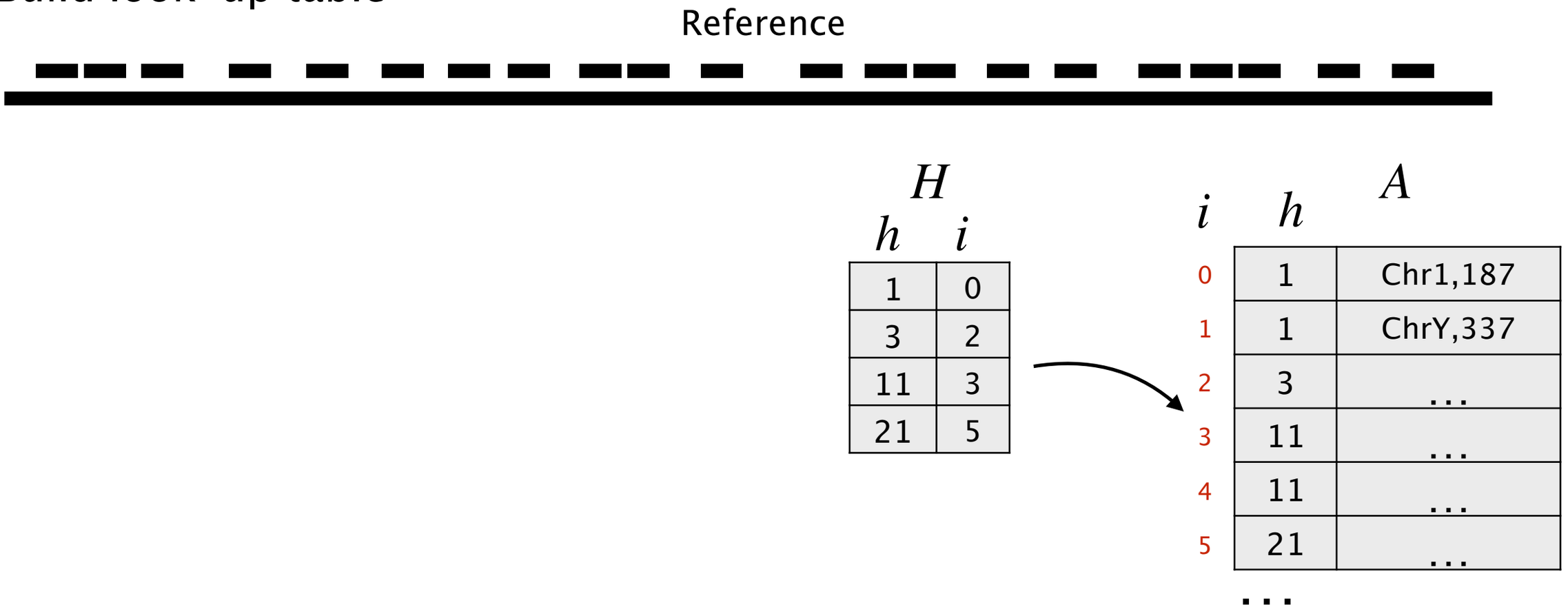


i	h	A
0	1	Chr1,187
1	1	ChrY,337
2	3	...
3	11	...
4	11	...
5	21	...
	...	

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



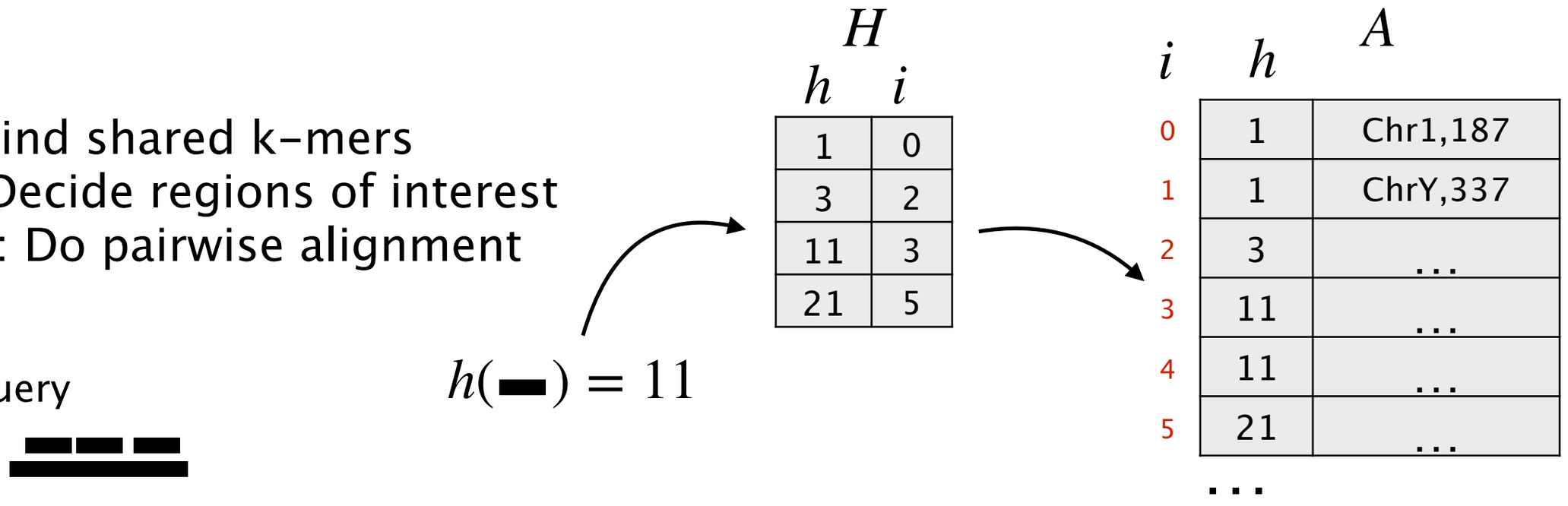
Hash-based seed-filter-extend

Indexing

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

Mapping

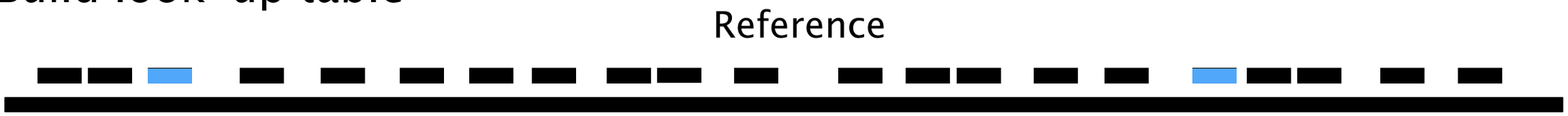
1. Seed: Find shared k-mers
2. Filter: Decide regions of interest
3. Extend: Do pairwise alignment



Hash-based seed-filter-extend

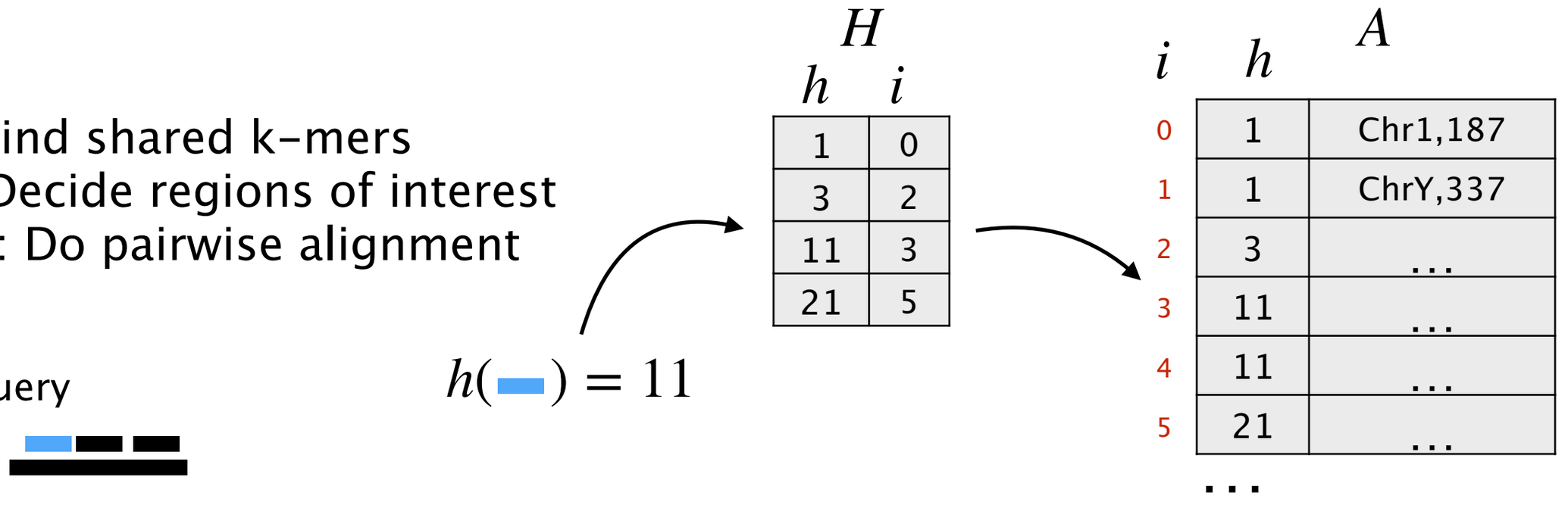
Indexing

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



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

Reference

Mapping

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

Query

H

h	i
1	0
3	2
11	3
21	5

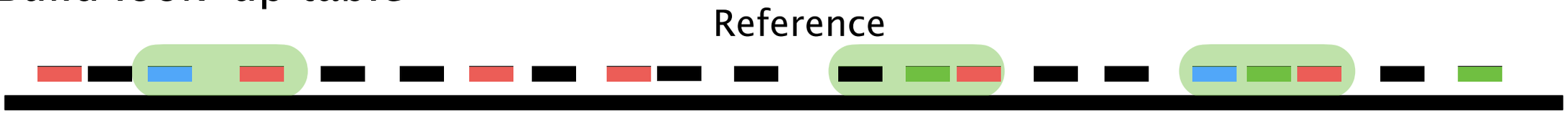
A

i	h	
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

h	i
1	0
3	2
11	3
21	5

A

i	h	
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

Reference

Mapping This talk

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

Query

H

h	i
1	0
3	2
11	3
21	5

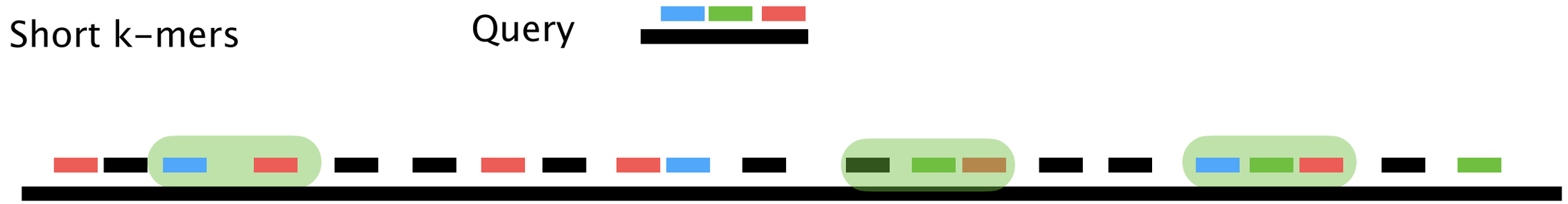
A

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

...

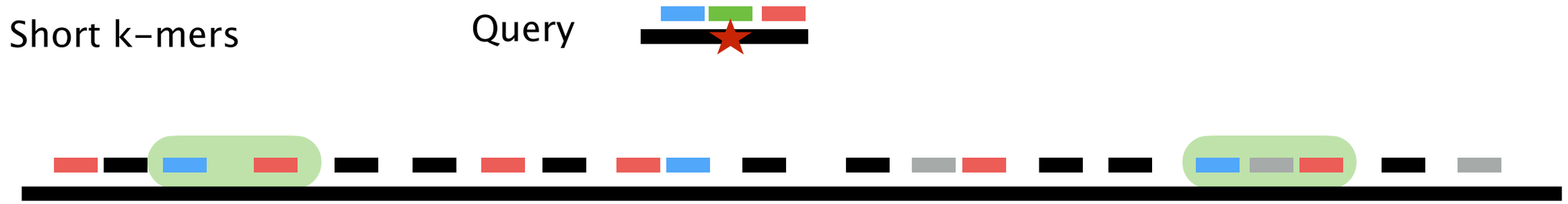
Sensitivity-specificity trade-off

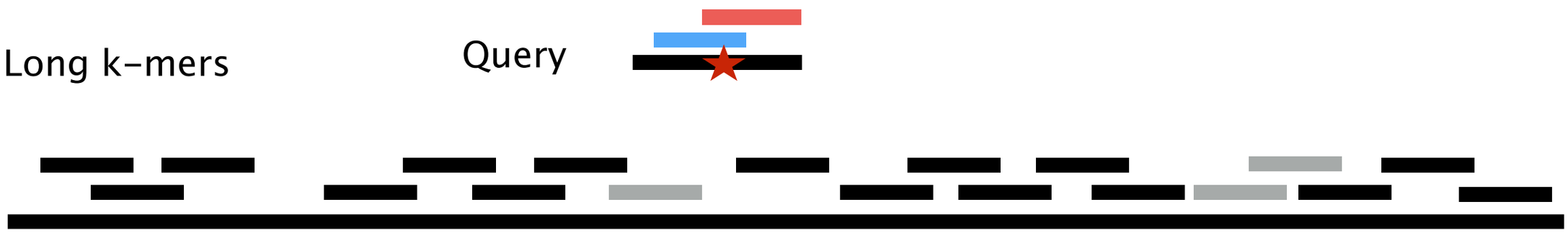
- Short k-mers: high sensitivity, low specificity
- Long k-mers: low sensitivity, high specificity



Sensitivity-specificity trade-off

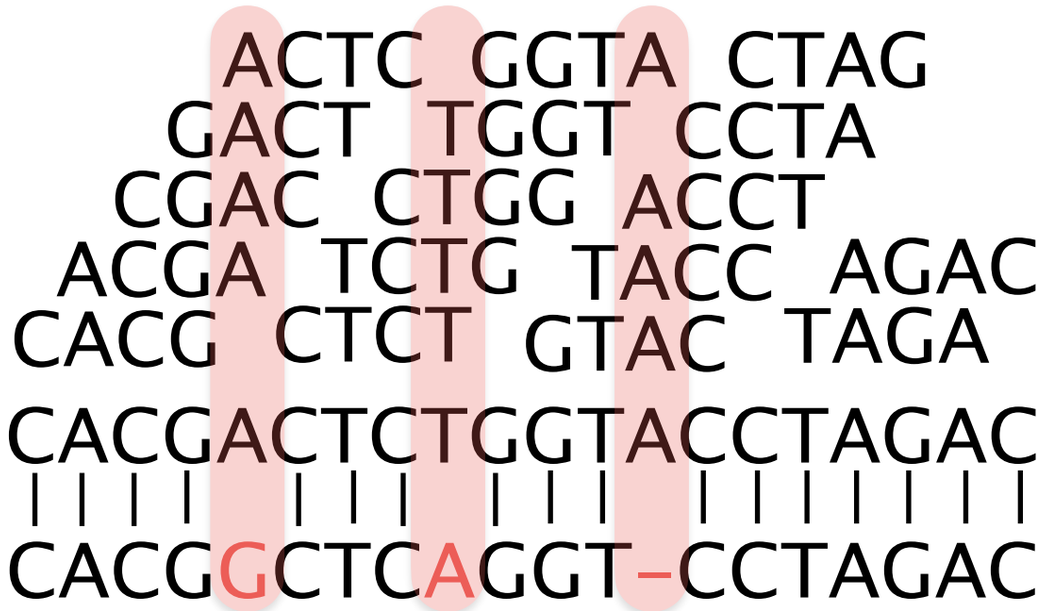
- Short k-mers: high sensitivity, low specificity
- Long k-mers: low sensitivity, high specificity





Approximate-matching (fuzzy) seeds

- Spaced k-mers[1]
- Vector seeds[2]
- Permutation techniques[3,4]
- Covering template families[5]
- SimHash[3] (BLEND[6])
- TensorSketch[7]
- SubSeqHash[8]
- Neighbouring minimizers[9]
- k-min-mers[10]
- Strobemers[11]

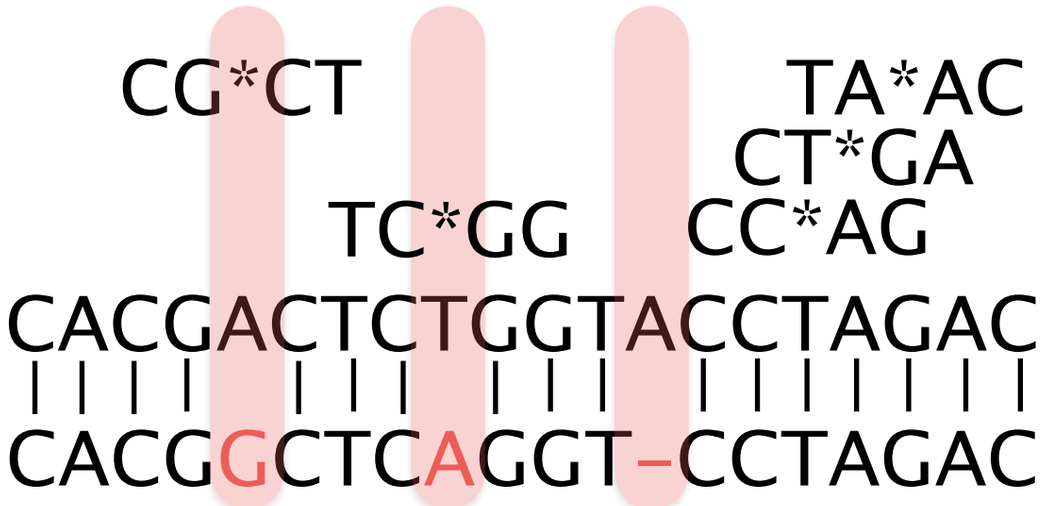


[1] Ma et al., 2002
[2] Brejová et al., 2003
[3] Charikar, 2002
[4] Lederman, 2013
[5] Giladi et al., 2010
[6] Firtina et al., 2021
[7] Joudaki et al., 2020
[8] Li et al., 2023
[9] Chin and Khalak, 2019
[10] Ekim, Berger, Chikhi, 2021
[11] Sahlin, 2021

Approximate-matching (fuzzy) seeds

- **Spaced k-mers**[1]
- Vector seeds[2]
- Permutation techniques[3,4]
- Covering template families[5]
- SimHash[3] (BLEND[6])
- TensorSketch[7]
- SubSeqHash[8]
- Neighbouring minimizers[9]
- k-min-mers[10]
- Strobemers[11]

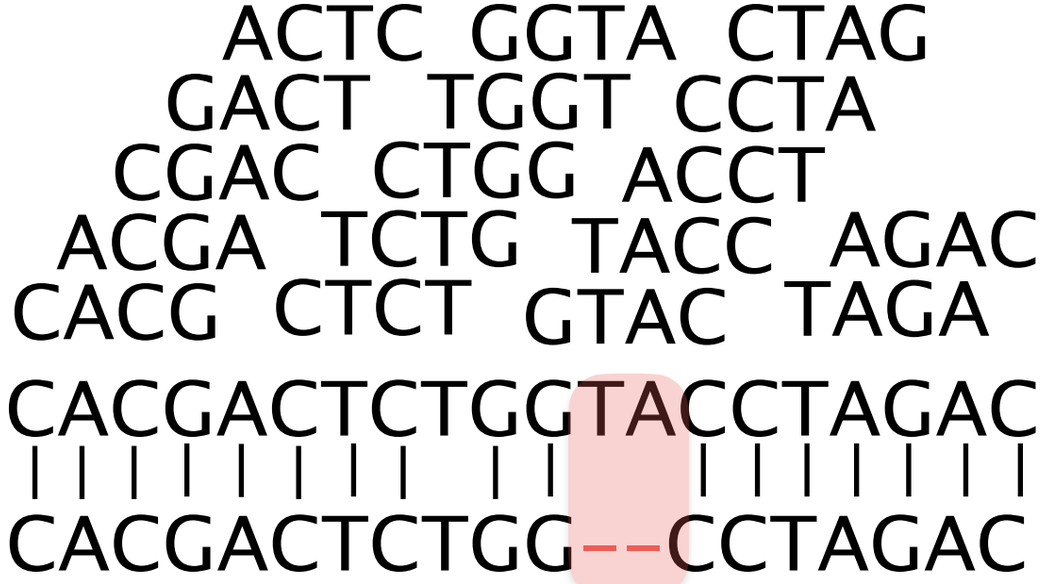
\+ Match over substitutions
\- Do not handle indels



[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

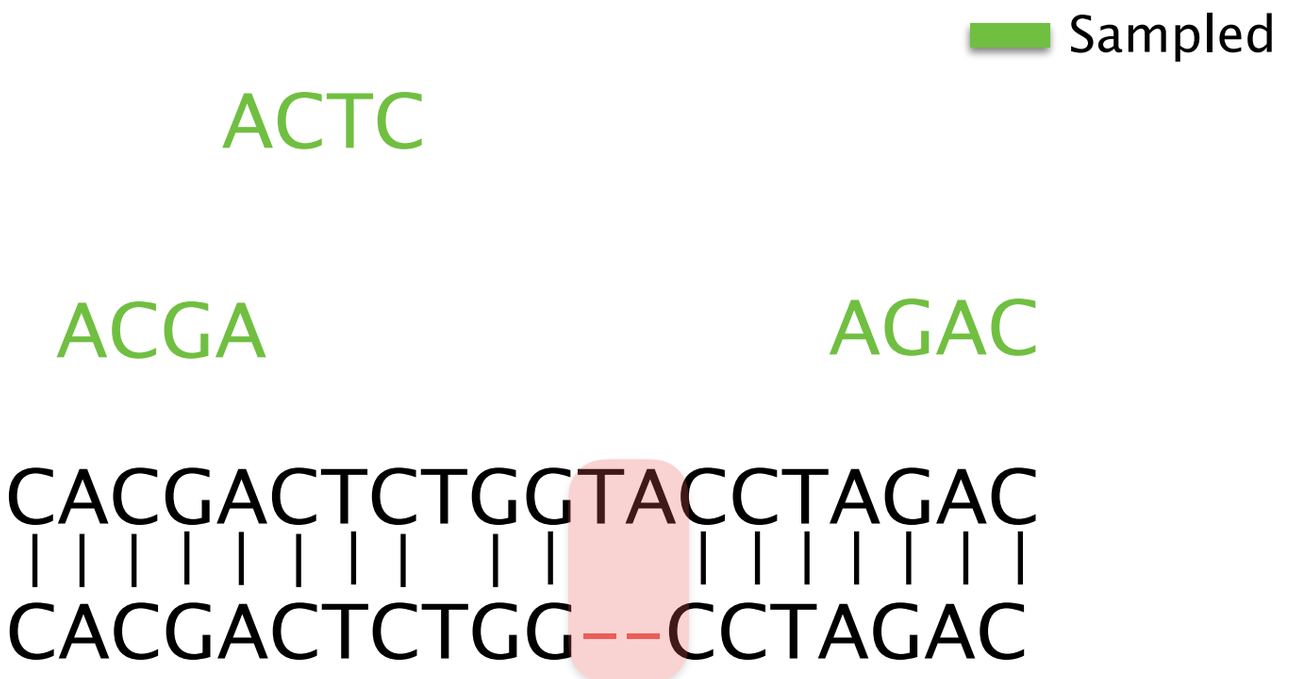
- Spaced k-mers[1]
- Vector seeds[2]
- Permutation techniques[3,4]
- Covering template families[5]
- SimHash[3] (BLEND[6])
- TensorSketch[7]
- SubSeqHash[8]
- **Neighbouring minimizers[9]**
- k-min-mers[10]
- Strobemers[11]



[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-mers[1]
- Vector seeds[2]
- Permutation techniques[3,4]
- Covering template families[5]
- SimHash[3] (BLEND[6])
- TensorSketch[7]
- SubSeqHash[8]
- **Neighbouring minimizers[9]**
- k-min-mers[10]
- Strobemers[11]

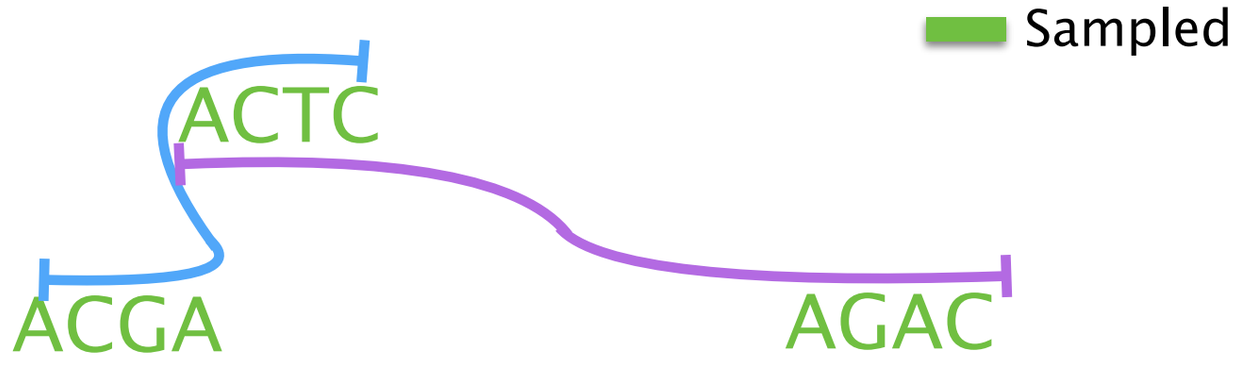


[1] Ma et al., 2002
[2] Brejová et al., 2003
[3] Charikar, 2002
[4] Lederman, 2013
[5] Giladi et al., 2010
[6] Firtina et al., 2021
[7] Joudaki et al., 2020
[8] Li et al., 2023
[9] Chin and Khalak, 2019
[10] Ekim, Berger, Chikhi, 2021
[11] Sahlin, 2021

Approximate-matching (fuzzy) seeds

- Spaced k-mers[1]
- Vector seeds[2]
- Permutation techniques[3,4]
- Covering template families[5]
- SimHash[3] (BLEND[6])
- TensorSketch[7]
- SubSeqHash[8]
- **Neighbouring minimizers[9]**
- k-min-mers[10]
- Strobemers[11]

\+ Match over subs. and indels
\- "k-mer redundancy"



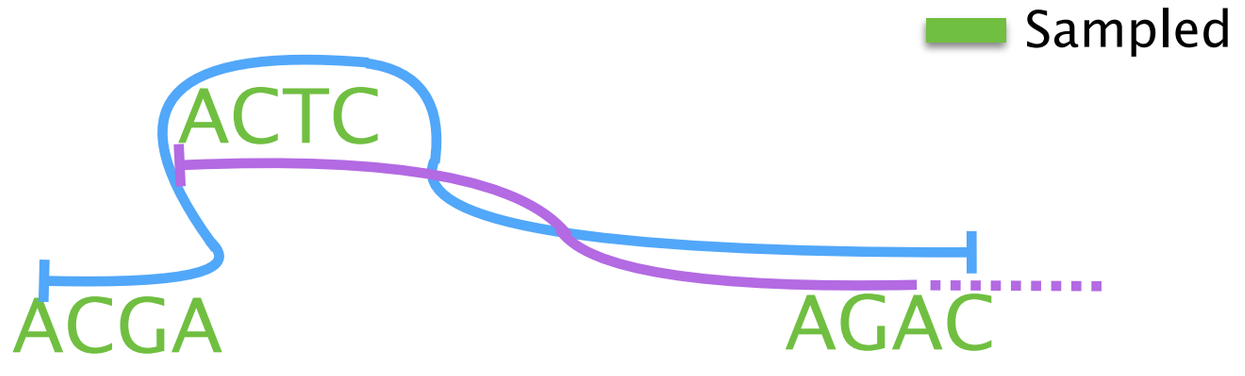

```
CACGACTCTGGTACCTAGAC
|||||||||  ||   |||||||
CACGACTCTGG--CCTAGAC
```

[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-mers[1]
- Vector seeds[2]
- Permutation techniques[3,4]
- Covering template families[5]
- SimHash[3] (BLEND[6])
- TensorSketch[7]
- SubSeqHash[8]
- Neighbouring minimizers[9]
- **k-min-mers[10]**
- Strobemers[11]

\+ Match over subs. and indels
\- "k-mer redundancy"




```
CACGACTCTGGTACCTAGAC
|||||||||  ||   |||||||
CACGACTCTGG--CCTAGAC
```

[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

Strobemers

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

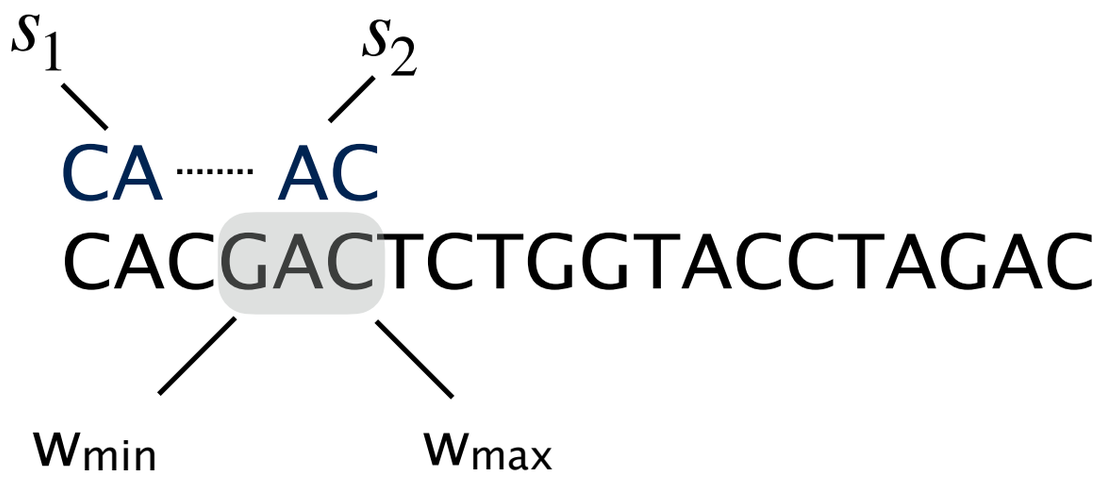
Sahlin, 2021, Genome Research

Strobemers

- Strobemers: n k-mers $s_1, s_2, \ldots, s_n$ (strobes) of length ℓ
  - s_1 is fixed
  - $s_2, \ldots, s_n$ chosen from downstream window [w_{min}, w_{max}]

Example

(n=2, ℓ=2, w_{min}=3, w_{max}=5)



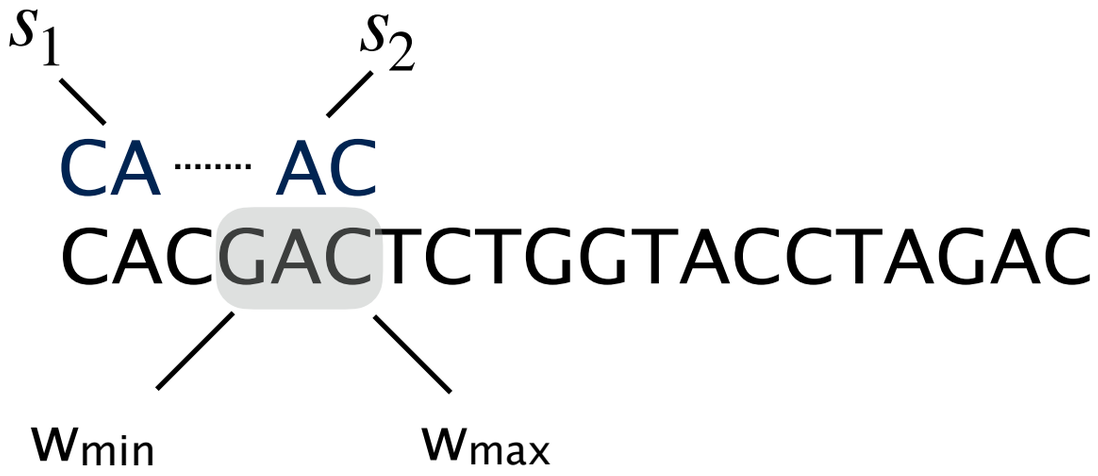
Sahlin, 2021, Genome Research

Strobemers

- Strobemers: n k-mers $s_1, s_2, \ldots, s_n$ (strobes) of length ℓ
  - s_1 is fixed
  - $s_2, \ldots, s_n$ chosen from downstream window [w_{min}, w_{max}]
  - Methods: minstrobes, **randstrobes**, hybridstrobes

Example

(n=2, ℓ=2, w_{min}=3, w_{max}=5)



Sahlin, 2021, Genome Research

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, ℓ=2, w_{min}=3, w_{max}=5)

s_1

CA

CACGACTCTGGTACCTAGAC

Sahlin, 2021, Genome Research

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

h(AA) = 0
h(AC) = 1
…
h(TT) = 15

q=5

CA
CACGACTGTGGTACCTAGAC

h(CA) = 4

Sahlin (2021)

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

CA
CACGACTGTGGTACCTAGAC

h(AA) = 0
h(AC) = 1
…
h(TT) = 15

q=5

h(CA) = 4

h(GA) = 8
h(AC) = 1
h(CT) = 7

Sahlin (2021)

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

CA
CACGACTGTGGTACCTAGAC

h(AA) = 0
h(AC) = 1
…
h(TT) = 15

q=5

h(CA) = 4

h(GA) = 8 (4 + 8) % 5 = 2
h(AC) = 1 (4 + 1) % 5 = 0
h(CT) = 7 (4 + 7) % 5 = 1

Sahlin (2021)

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

CA…… AC
CACGACTGTGGTACCTAGAC

h(AA) = 0
h(AC) = 1
…
h(TT) = 15

q=5

h(CA) = 4

h(GA) = 8 (4 + 8) % 5 = 2
h(AC) = 1 (4 + 1) % 5 = 0
h(CT) = 7 (4 + 7) % 5 = 1

Sahlin (2021)

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

AC
CA····· AC
CACGACTGTGGTACCTAGAC

h(AA) = 0
h(AC) = 1
…
h(TT) = 15

q=5

h(AC) = 1

h(AC) = 1 (1 + 1) % 5 = 2
h(CT) = 7 (1 + 7) % 5 = 3
h(TG) = 14 (1 + 14) % 5 = 0

Sahlin (2021)

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

AC ····· TG
CA····· AC
CACGACTGTGGTACCTAGAC

h(AA) = 0
h(AC) = 1
…
h(TT) = 15

q=5

h(AC) = 1

h(AC) = 1 (1 + 1) % 5 = 2
h(CT) = 7 (1 + 7) % 5 = 3
h(TG) = 14 (1 + 14) % 5 = 0

Sahlin (2021)

Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)

CG ·· CT
AC ····· TG
CA····· AC
CACGACTGTGGTACCTAGAC

Sahlin (2021)

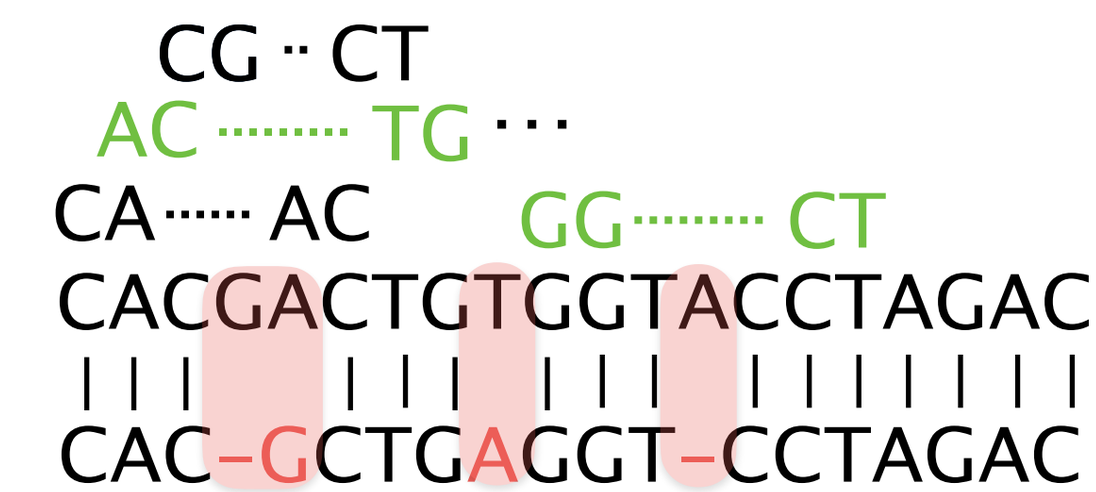
Randstrobes

- 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, ℓ=2, w_{min}=3, w_{max}=5)



Sahlin, 2021, Genome Research

Randstrobes

- 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=2, ℓ=2, w_{min}=3, w_{max}=5)

CA ····· AC ····· GG
CACGACTGTGGTACCTAGAC

Sahlin (2021)

Randstrobes

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

Example

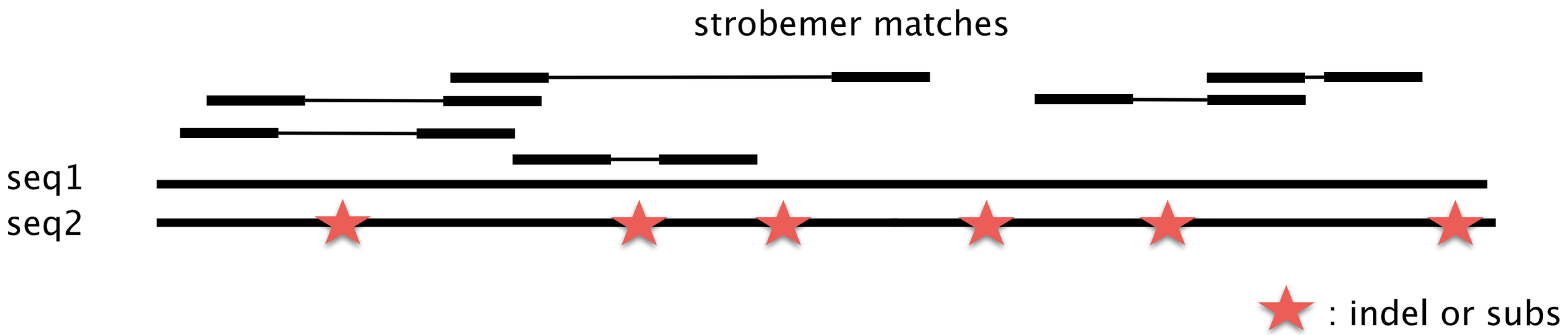
(n=2, ℓ=2, w_{min}=3, w_{max}=5)

CA ····· AC ····· GG
CACGACTGTGGTACCTAGAC

Sahlin (2021)

Measuring sensitivity

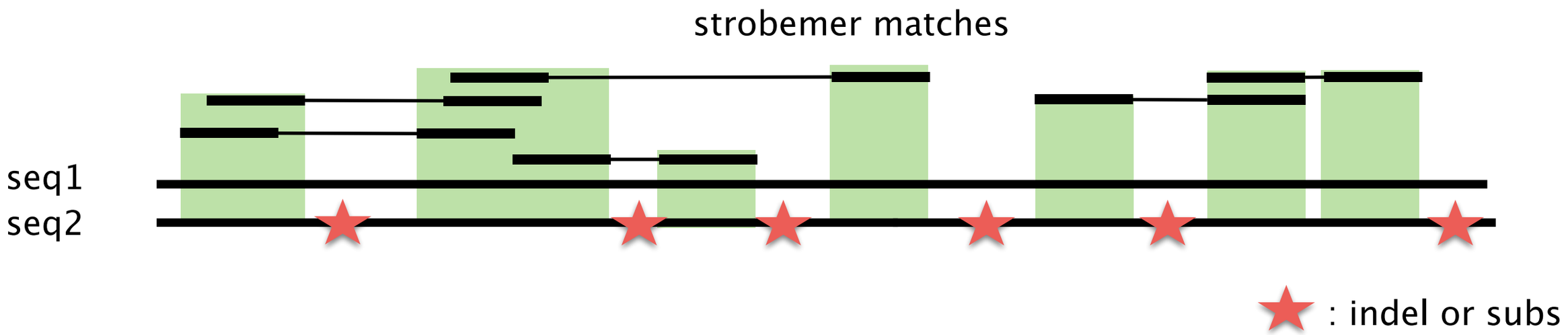
- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



Sahlin, 2021, Genome Research

Measuring sensitivity

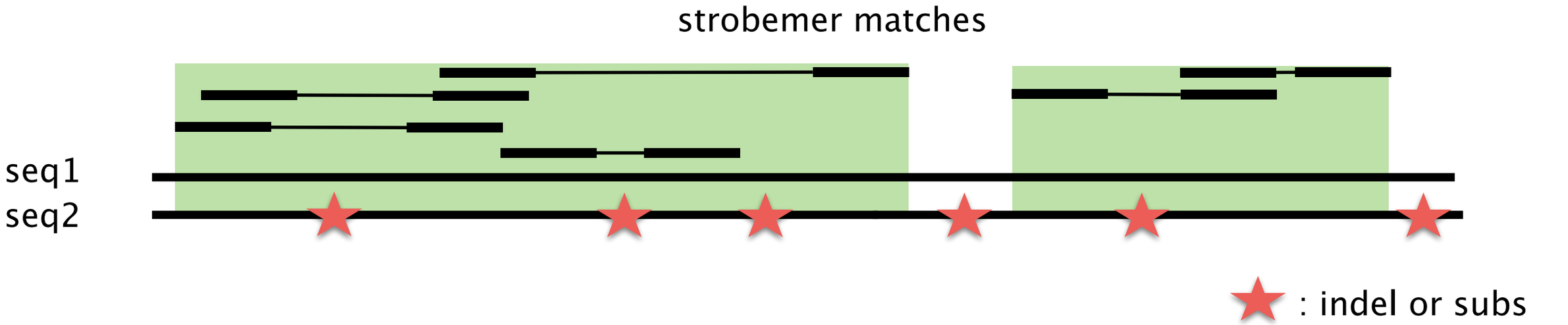
- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



Sahlin, 2021, Genome Research

Measuring sensitivity

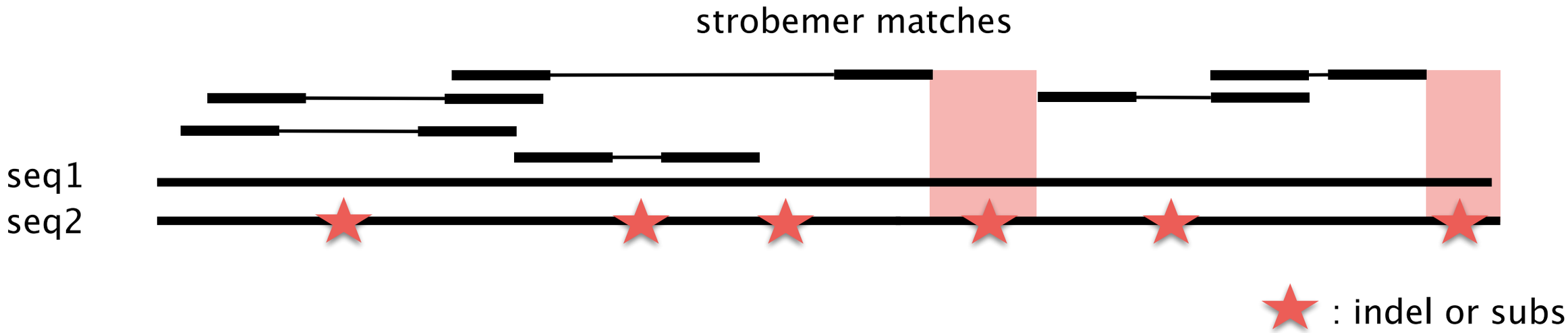
- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



Sahlin, 2021, Genome Research

Measuring sensitivity

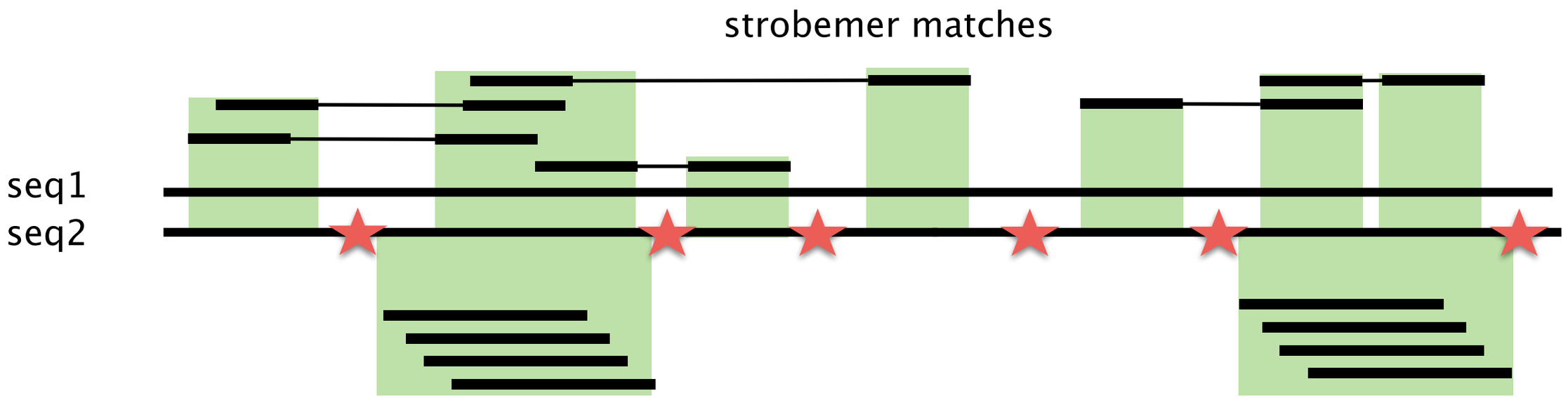
- Number of matches (m)
- Sequence coverage (sc)
- Match coverage (mc)
- Match gap E-size (E)



Sahlin, 2021, Genome Research

Randstrobes are sensitive

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



Sahlin, 2021, Genome Research

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?

Sahlin, 2021, Genome Research
Sahlin, 2022, Genome Biology

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?

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

Expected seed count

Total seeds

Distinct seeds

Count seed i

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

Sahlin, 2021, Genome Research
Sahlin, 2022, Genome Biology

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

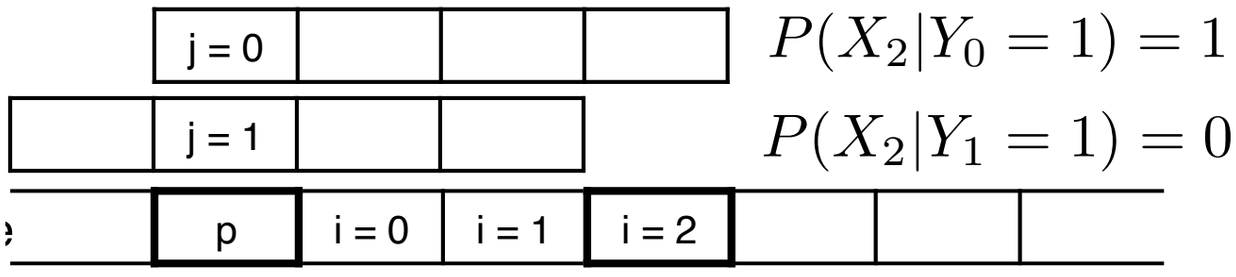
Conclusion: Comparable specificity for reasonable parameter choices

Sahlin, 2021, Genome Research
Sahlin, 2022, Genome Biology

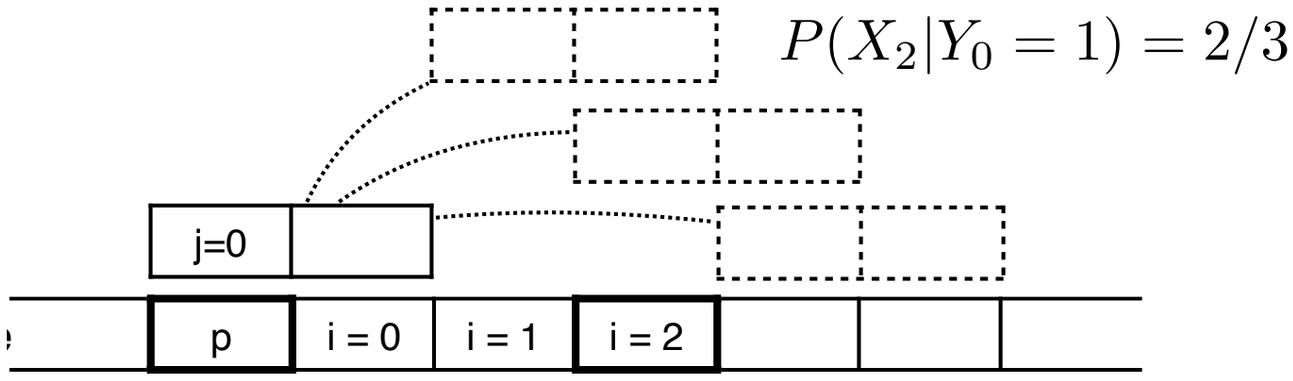
Why are strobemers sensitive?

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

k-mers (4)



randstrobes (2,2,2,4)

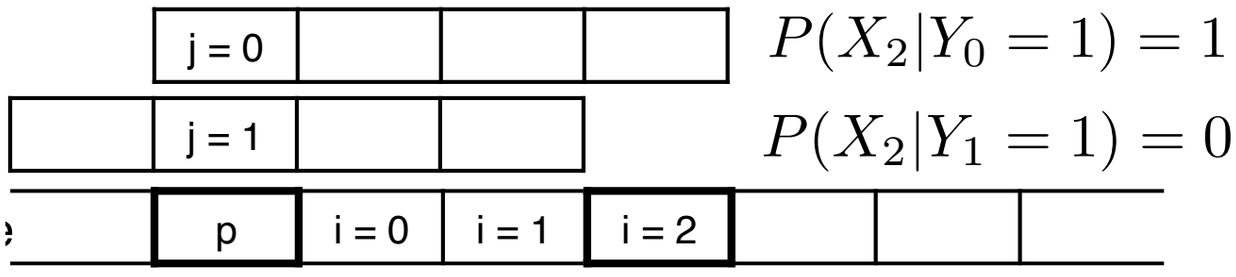


Maier and Sahlin 2023, Genome Research

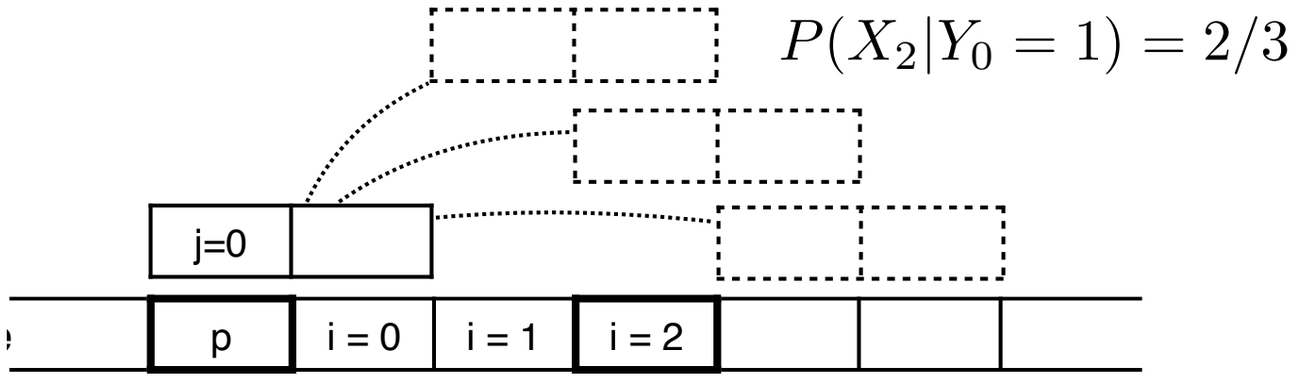
Why are strobemers sensitive?

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

k-mers (4)



randstrobes (2,2,2,4)



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

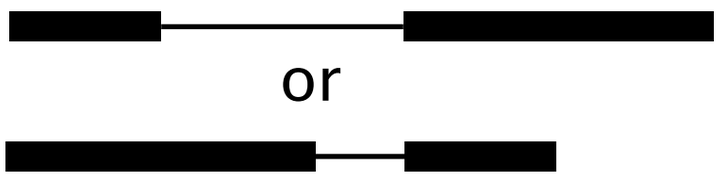
Maier and Sahlin 2023, Genome Research

Can we design seeds with higher entropy?

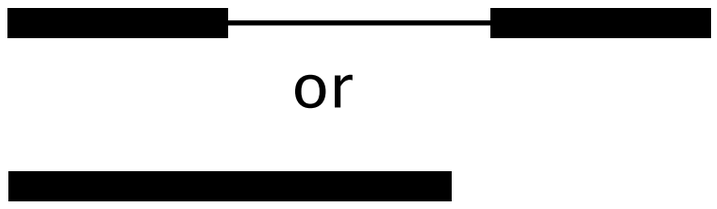
- Altstrobes: Altering the size of first and second strobe ($s_1 + s_2 = k$)
- Mixedstrobes: Mixing k-mers and randstrobes
- Multistrobes: Selecting strobe size from multiple lengths (still with $s_1 + s_2 = k$)

Randstrobes

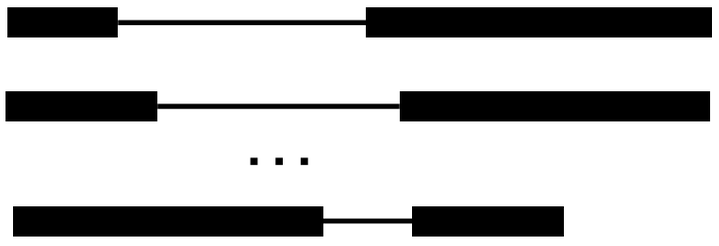
Altstrobes



Mixedstrobes



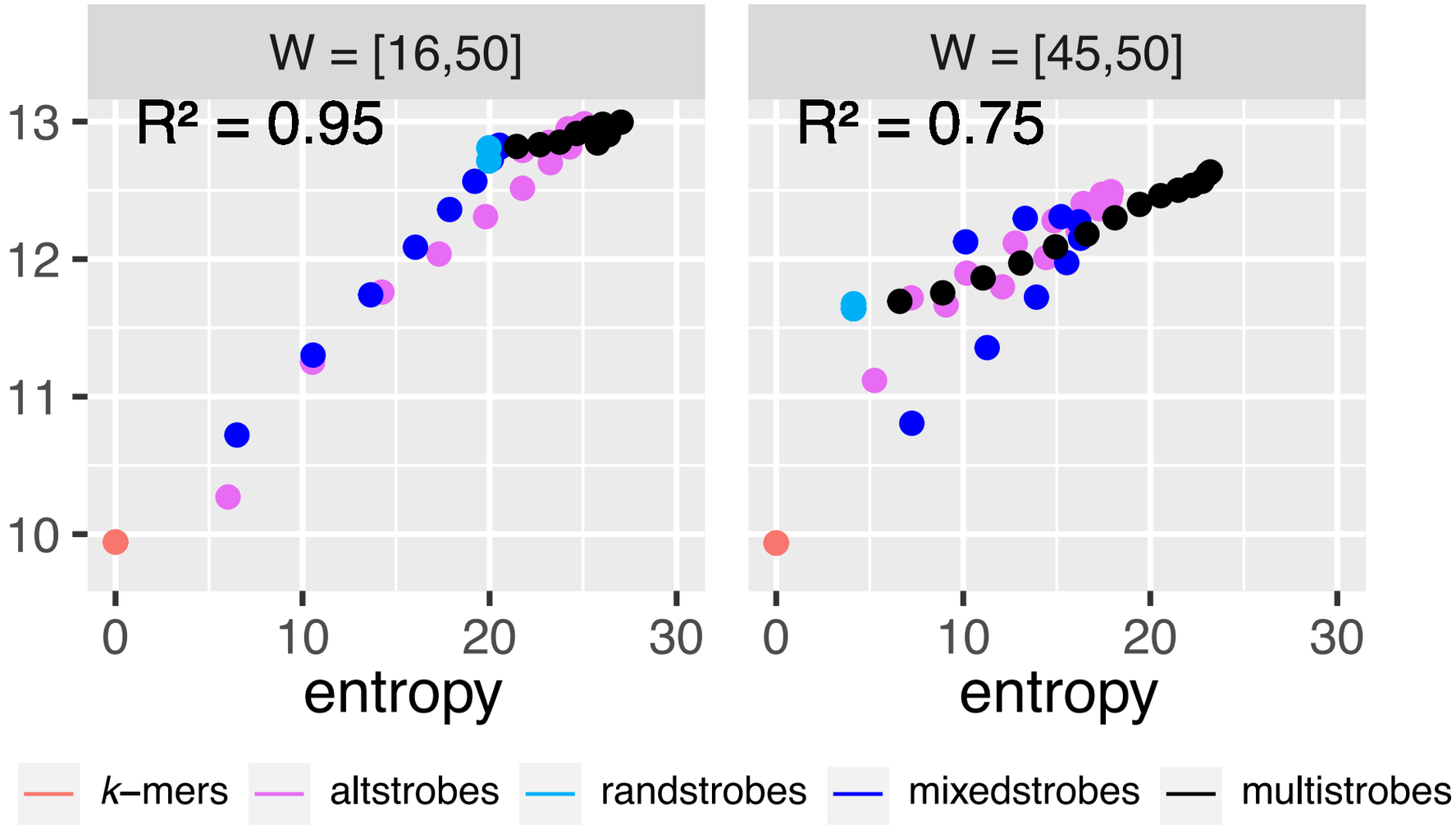
Multistrobes



Maier and Sahlin 2023, Genome Research

Higher entropy leads to higher sensitivity

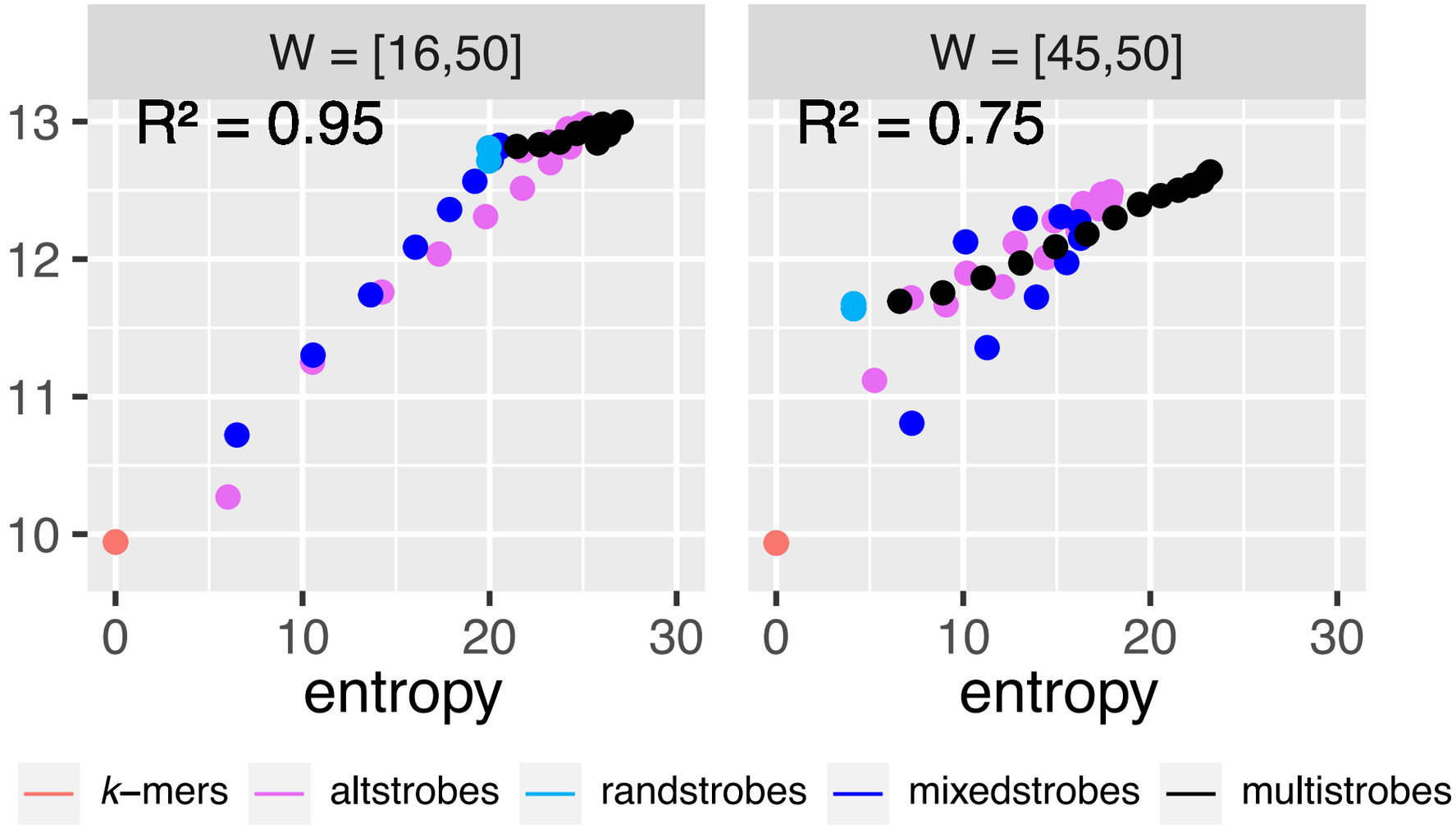
$$\sum_{\epsilon} P(M \mid \epsilon)$$ ~~sensitivity~~



Maier and Sahlin 2023, Genome Research

Higher entropy leads to higher sensitivity

$$\sum_{\epsilon} P(M \mid \epsilon)$$ ~~sensitivity~~



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

Maier and Sahlin 2023, Genome Research

Runtime: Probably not a bottleneck

		Human Chr1
k-mers	30bp	**12,4**
randstrobes	(2,15,16,40)	19,1
randstrobes	(2,15,16,100)	19,4
randstrobes	(3,10,11,40)	25,1
randstrobes	(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 $(h(s_1) + h(s_2)) \& q$**

Sahlin, 2021, Genome Research

Can we optimise the implementation?

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

Karami et al. 2024, Bioinformatics

Can we optimise the implementation?

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

Bitwise AND with bitmask (Wei Shen)

Karami et al. 2024, Bioinformatics

Can we optimise the implementation?

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

Bitwise XOR (Lidong Guo ,Giulio Pibiri)

Karami et al. 2024, Bioinformatics

Can we optimise the implementation?

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

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 $f(s_1, s_2)$

Karami et al. 2024, Bioinformatics

Can we optimise the implementation?

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

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

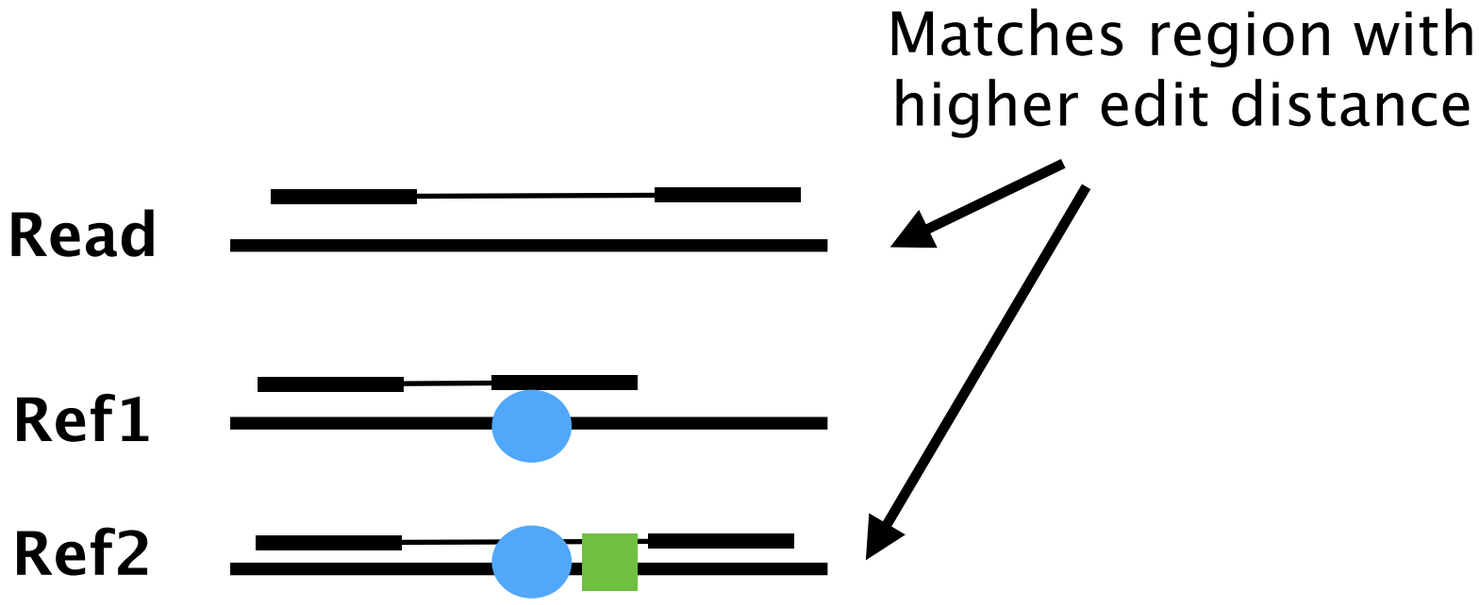
- Fast
- Maximize entropy of sampled strobe in window

Results

- Link method influences both time and randomness

Karami et al. 2024, Bioinformatics

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

- RNAbloom2[1]: Reference-free long-read transcriptome assembly
- BLEND[2]: Long-read overlap detection
- Symbiont-screener[3]: Reference-free long read phasing based on “markers”
- SAKE[4]: Identify all k-mers in a genome from long reads

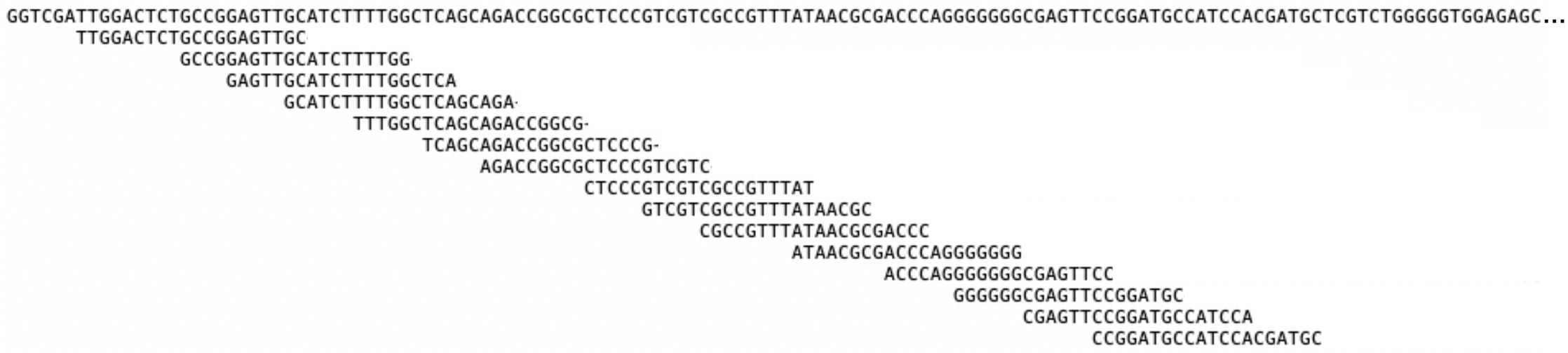
Our work

- strobealign[5]: Short-read alignment.

[1]Nip et al, Nat Commun, 2023
[2]Firtina et al., NAR genom. bioinform., 2023
[3]Xu et al., Front. mar. sci., 2023.
[4]Leinonen and Salmela, Plos One, 2023.
[5]Sahlin, Genome Biology, 2022.

Short-read mapping with strobealign

- strobealign[1] combines syncmers[2] and randstrobes[3]
- Main idea: Can afford to use longer seeds

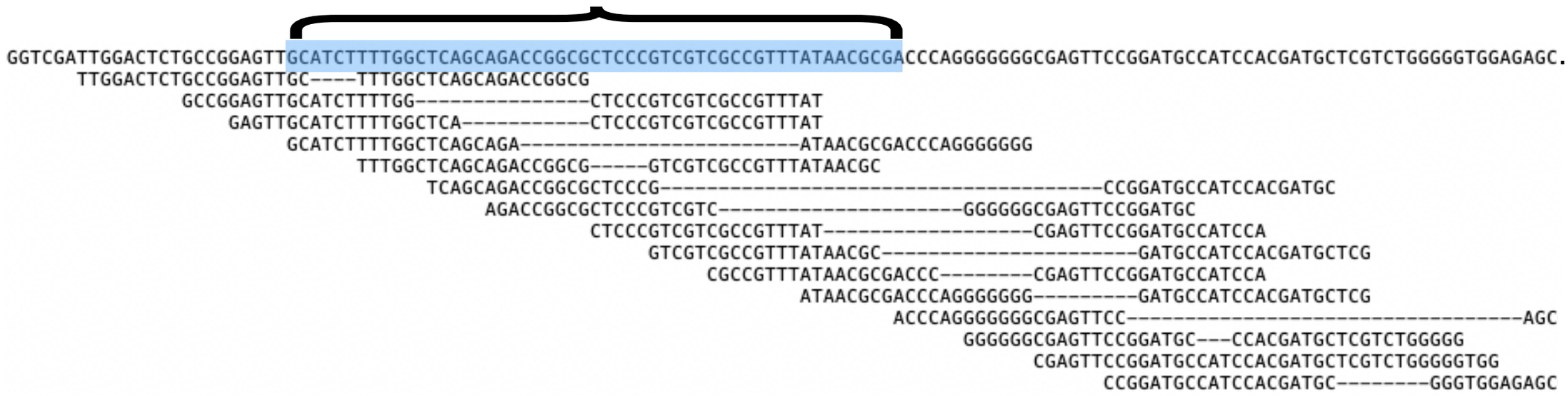


[1]Sahlin, Genome Biology, 2022.
[2] Edgar, PeerJ, 2020
[3] Sahlin, Genome Research. 2021

Short-read mapping with strobealign

- strobealign[1] combines syncmers[2] and randstrobes[3]
- Main idea: Can afford to use longer seeds

Strobe from window of syncmers instead of k-mers

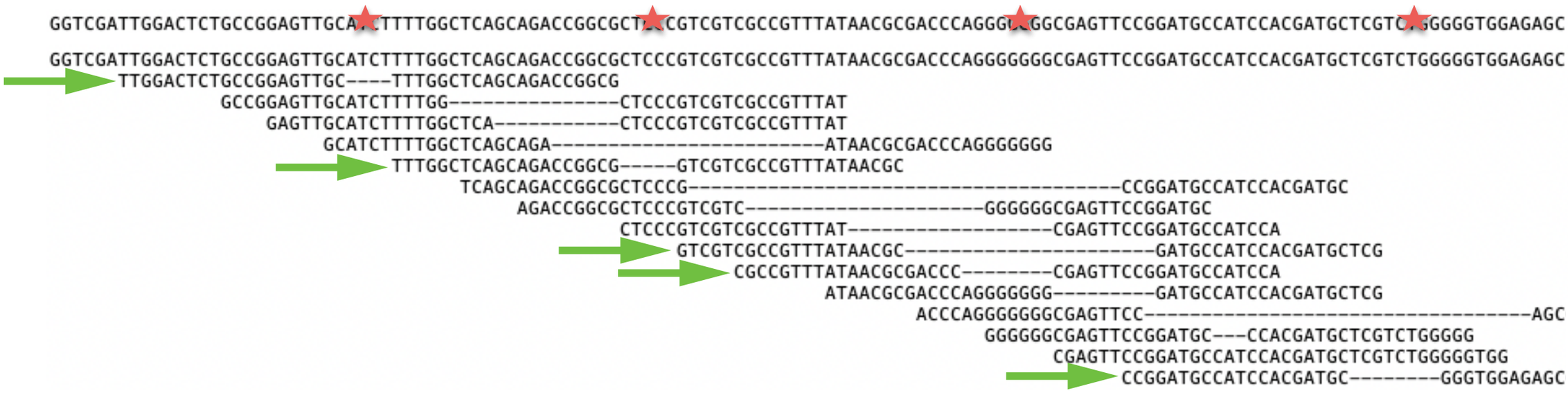


[1]Sahlin, Genome Biology, 2022.
[2] Edgar, PeerJ, 2020
[3] Sahlin, Genome Research. 2021

Short-read mapping with strobealign

- strobealign[1] combines syncmers[2] and randstrobes[3]
- Main idea: Can afford to use longer seeds

Reference

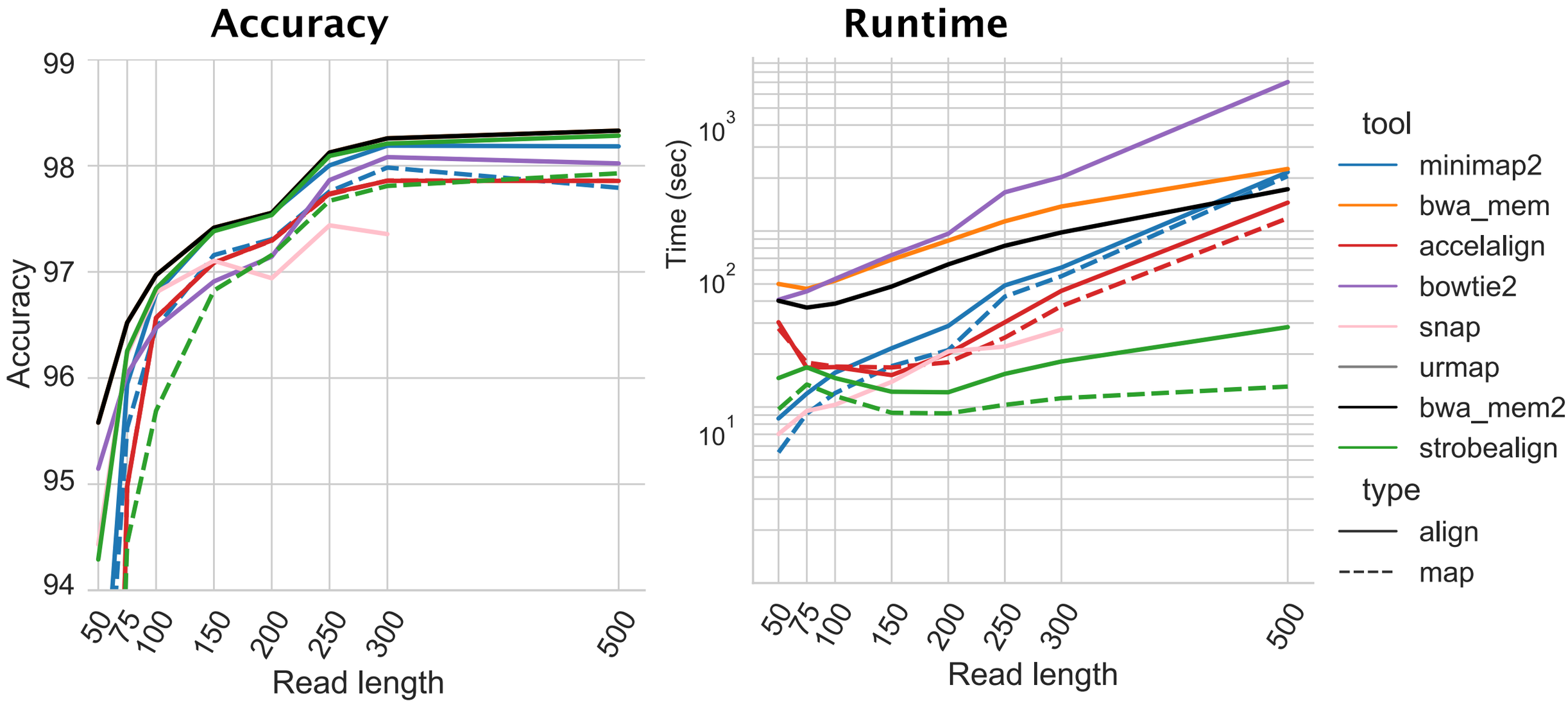


Chance to match

[1]Sahlin, Genome Biology, 2022.
[2] Edgar, PeerJ, 2020
[3] Sahlin, Genome Research. 2021

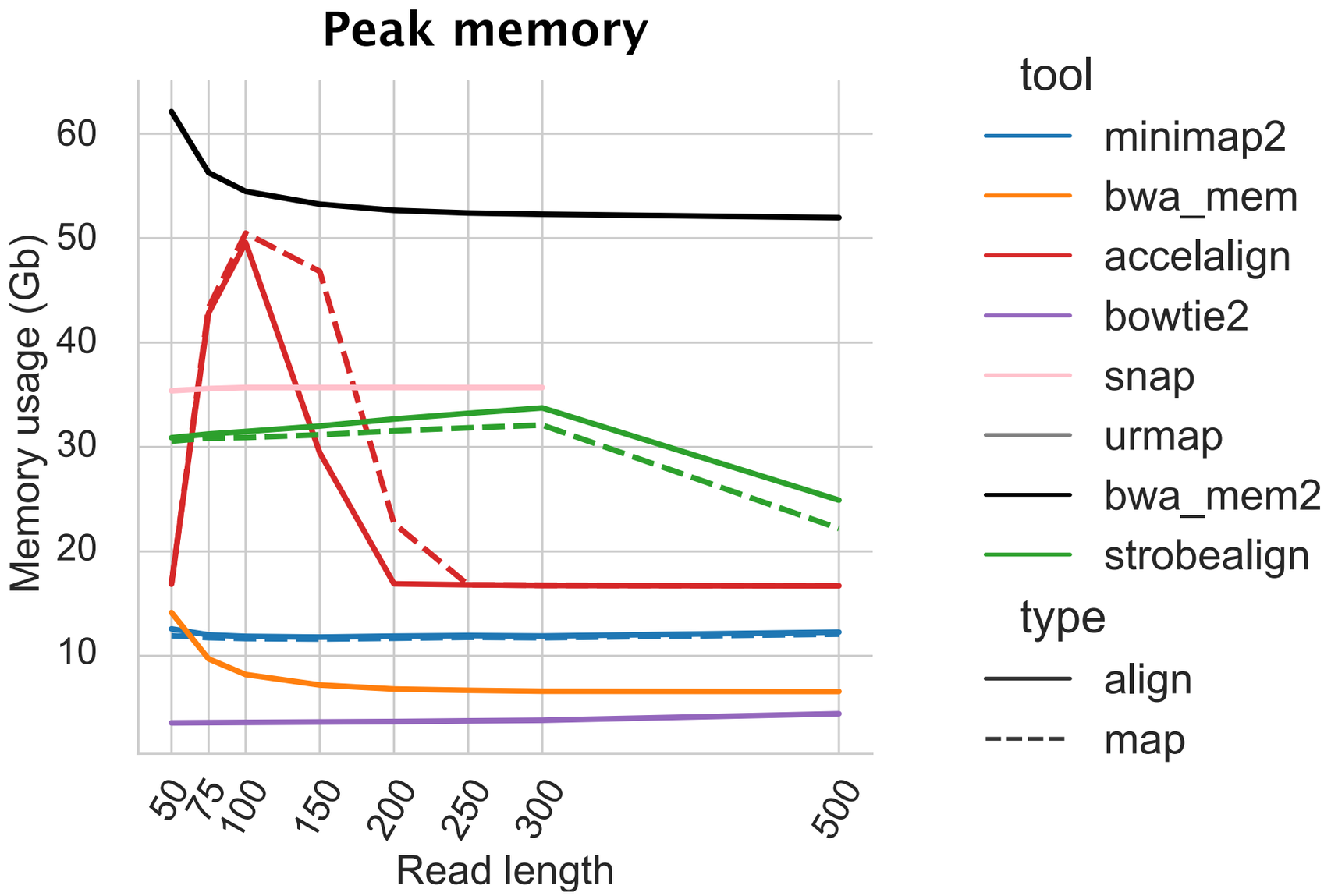
Strobealign has high accuracy ($\geq 150bp$) and fast

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



Sahlin, Genome Biology, 2022

But high memory consumption!



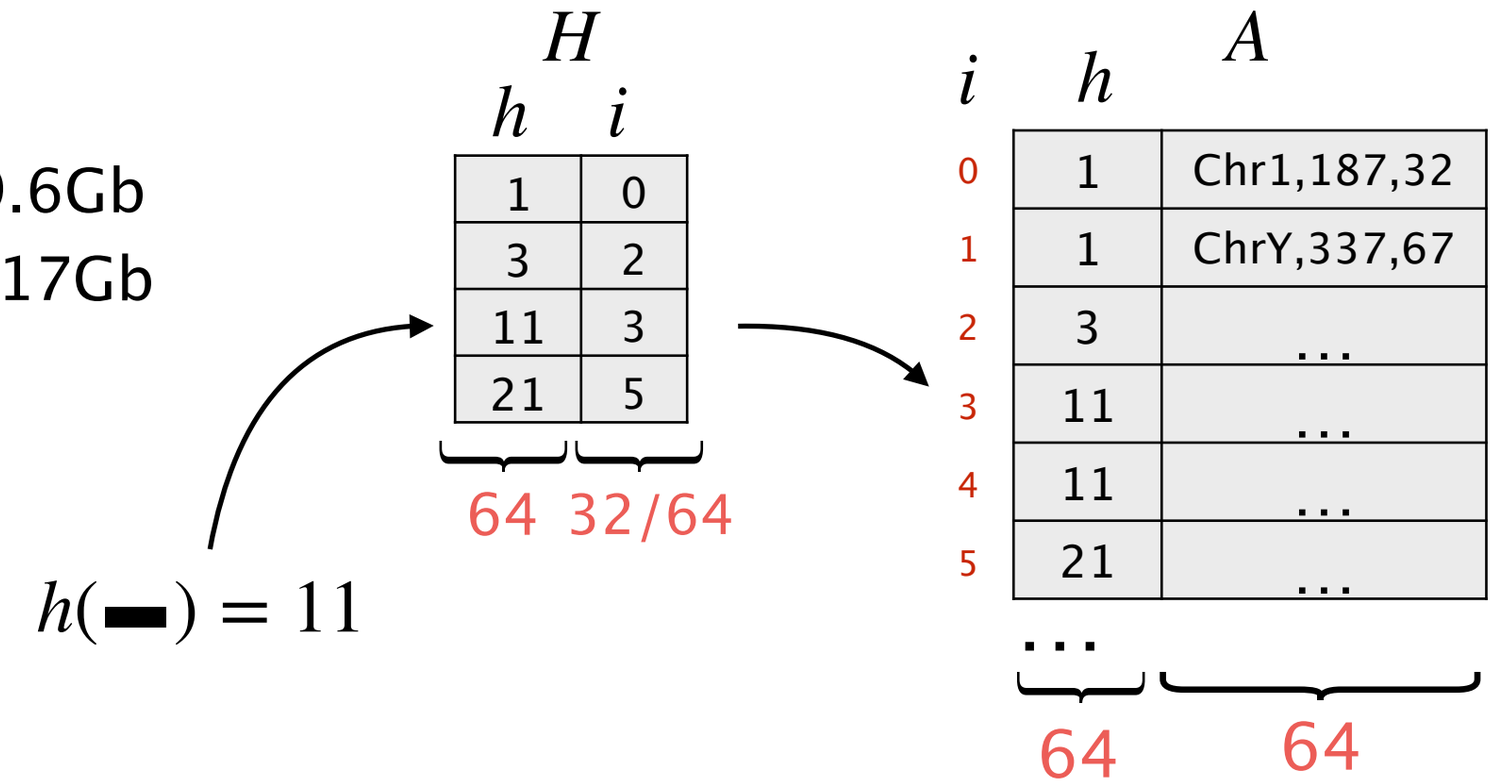
Sahlin, Genome Biology, 2022

What is consuming memory?

Reference

Human

- Genome: 3Gb
- A: 3B*(1/5)*128 bits = 9.6Gb
- H= load factor etc…? ≈ 17Gb



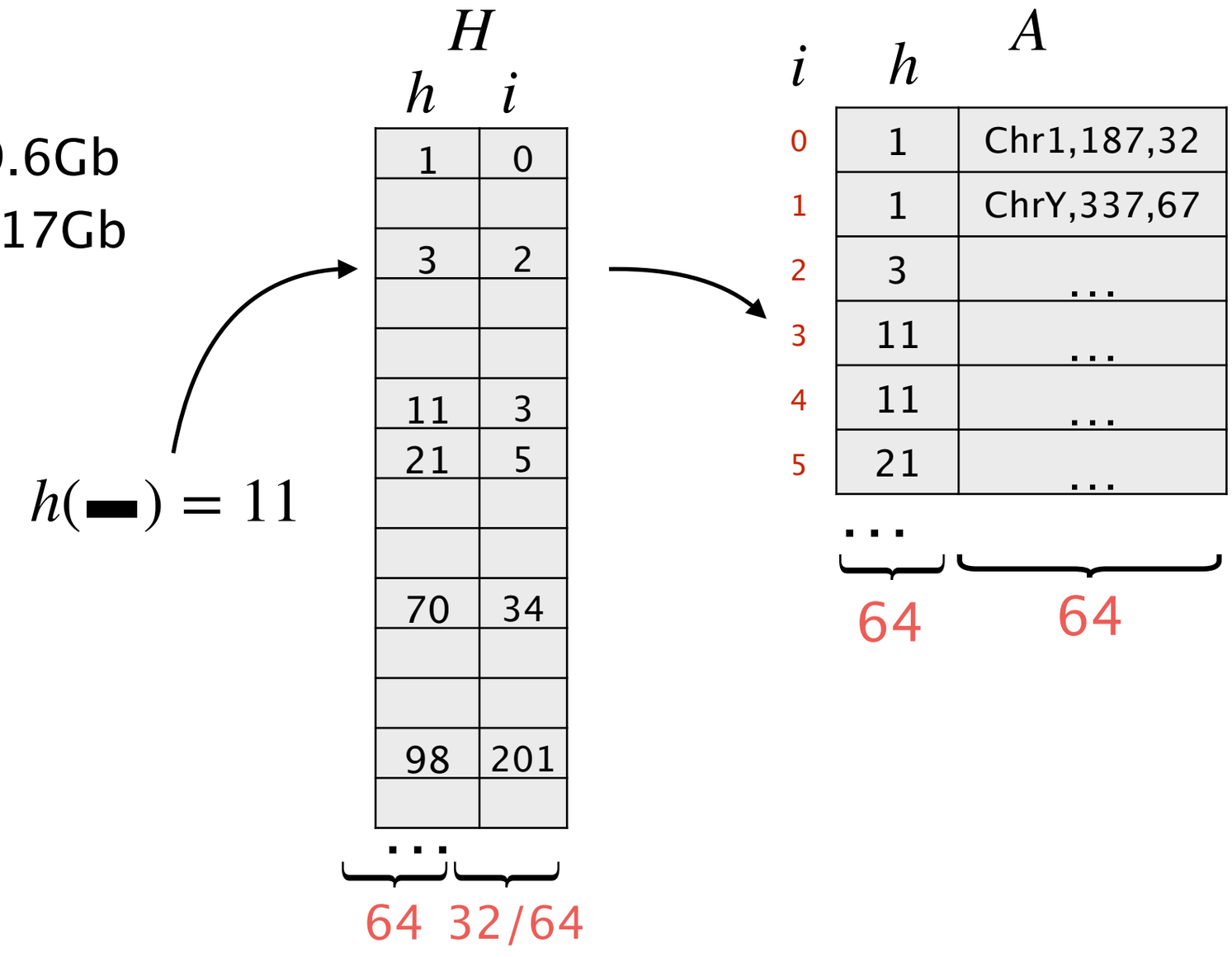
*Pan et al., forthcoming study

What is consuming memory?

Reference

Human

- Genome: 3Gb
- A: 3B*(1/5)*128 bits = 9.6Gb
- H= load factor etc…? ≈ 17Gb



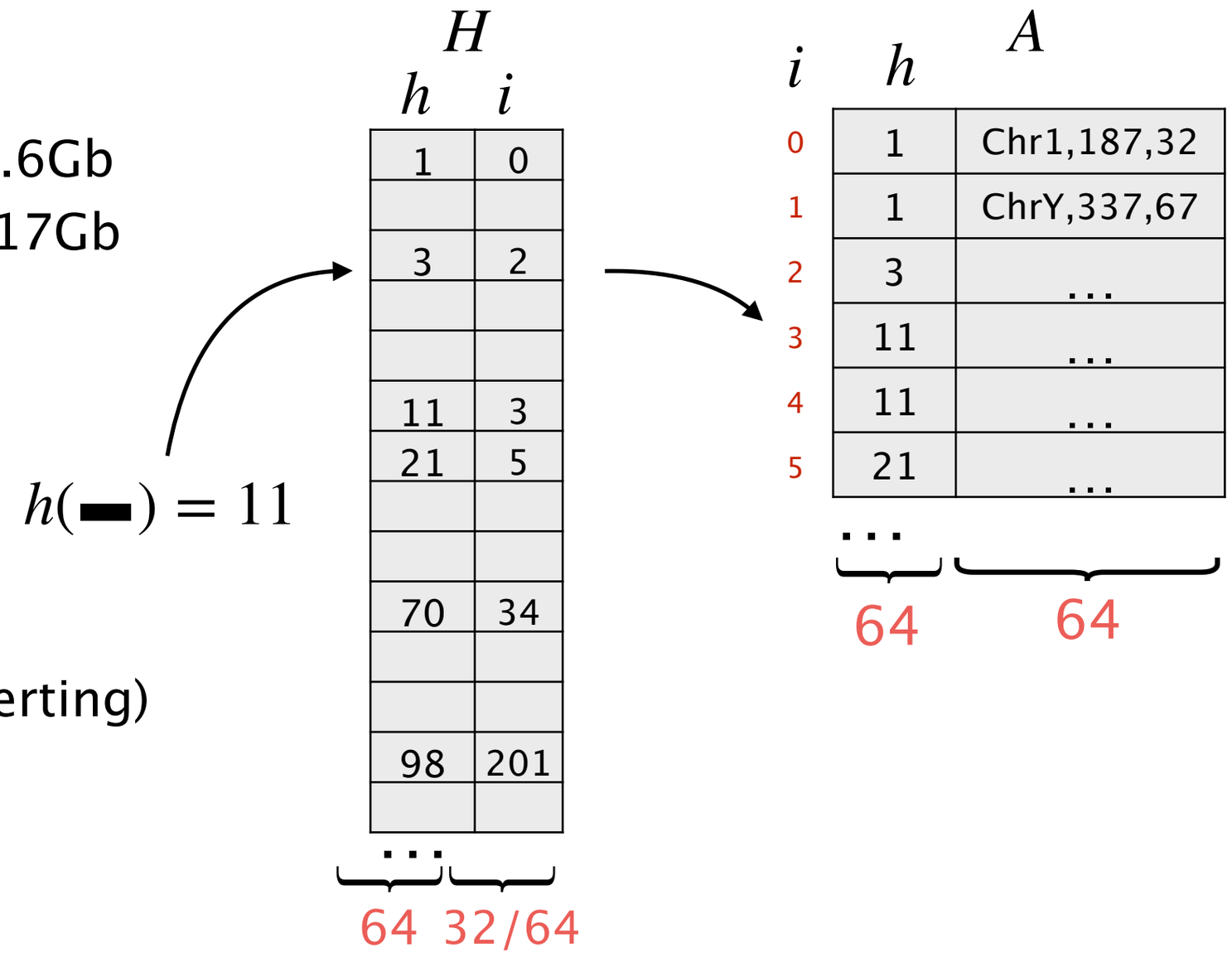
*Pan et al., forthcoming study

What is consuming memory?

Reference

Human

- Genome: 3Gb
- A: 3B*(1/5)*128 bits = 9.6Gb
- H= load factor etc…? ≈ 17Gb



Idea (Cred. Luis Coelho)

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

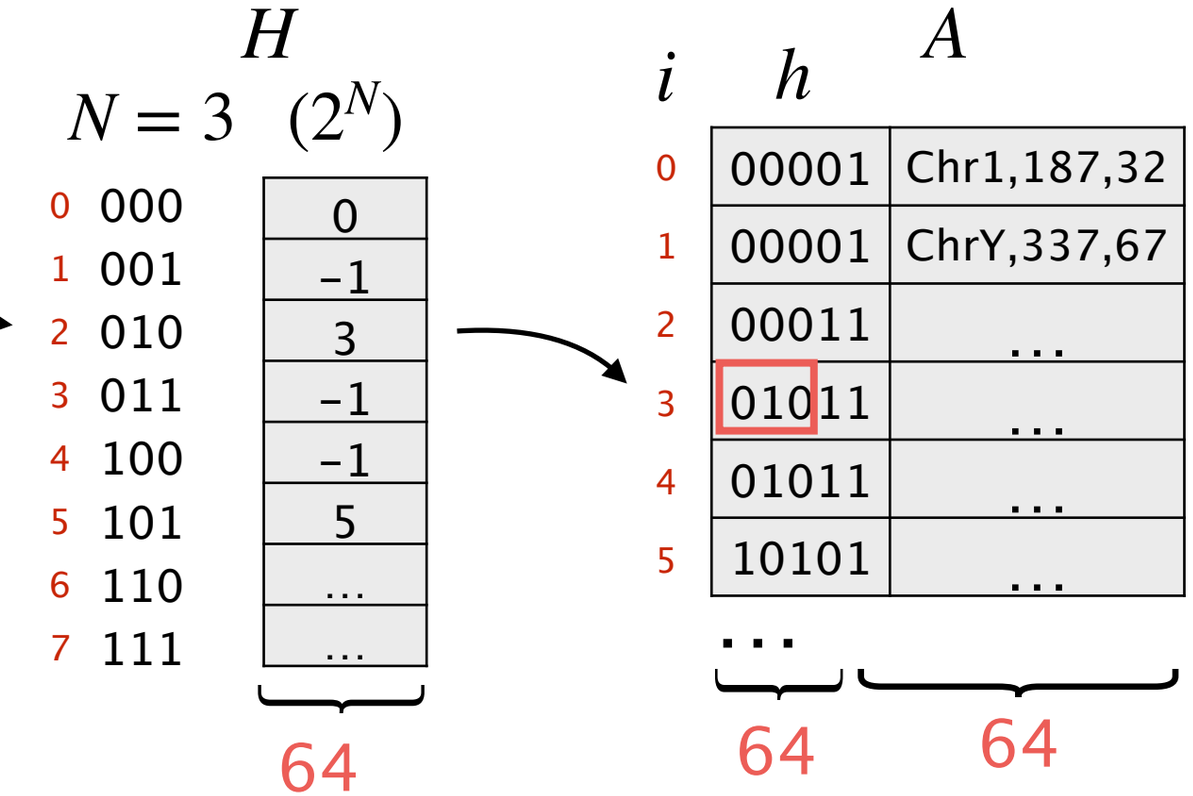
*Pan et al., forthcoming study

What is consuming memory?

Reference

Human

- Genome: 3Gb
- A: 3B*(1/5)*128 bits = 9.6Gb
- H= load factor etc…? ≈ 17Gb



$h(\blacksquare) = 11 = 01011$

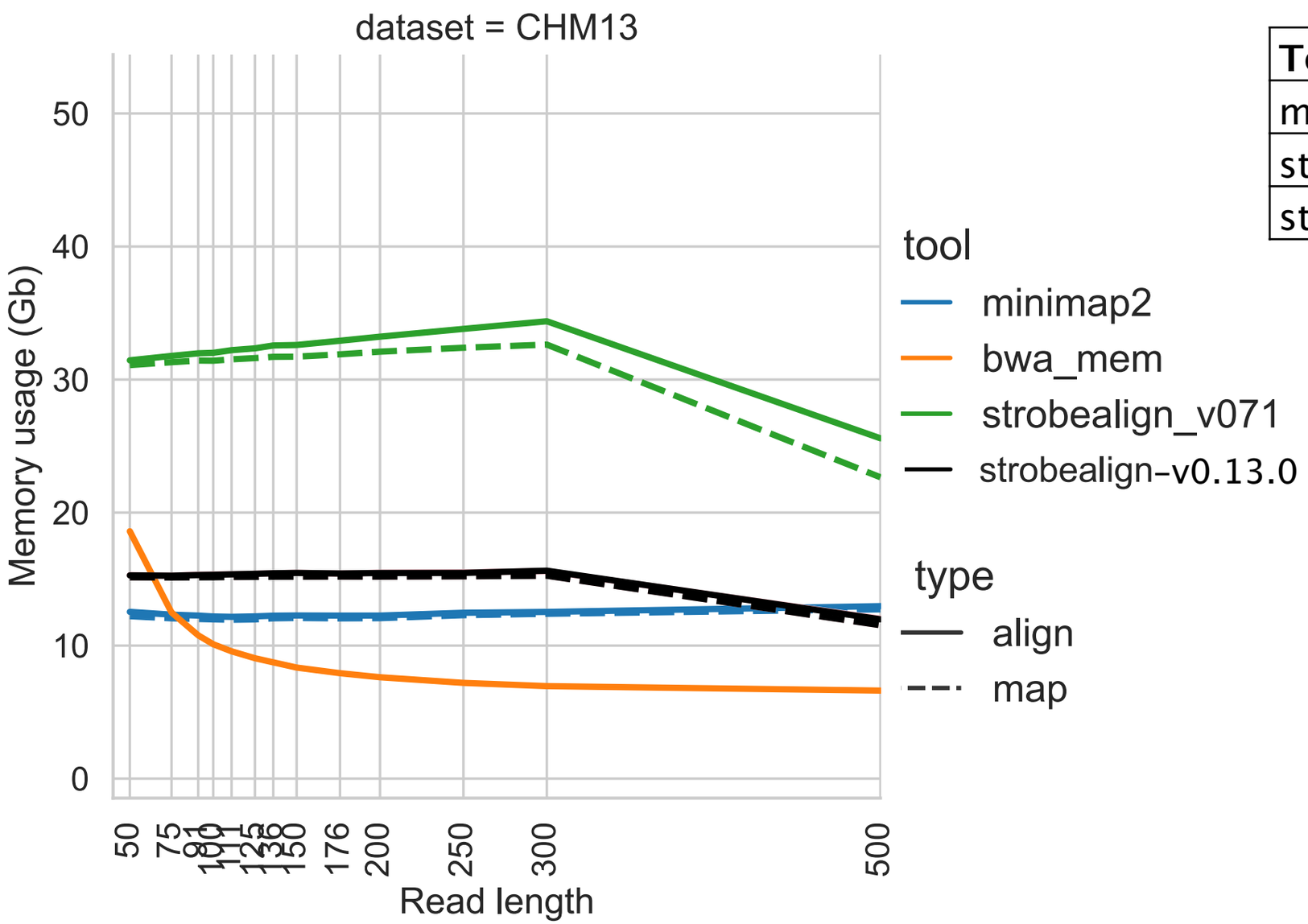
Idea (Cred. Luis Coelho)

- A is static (no deleting/inserting)
- Hash table not needed
- Use prefix lookup vector!
- $N = 28$ -> 268M slots -> ~2Gb

*Pan et al., forthcoming study

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

Memory



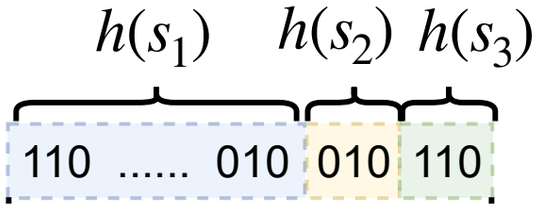
Time

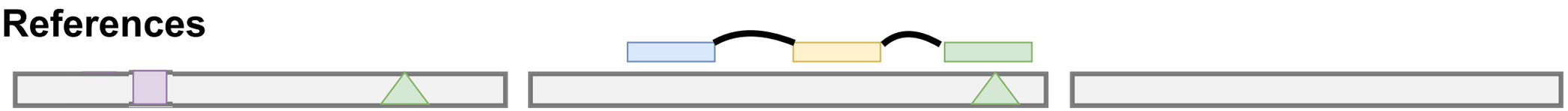
Tool (1 thread)	time
minimap2	191s
strobealign-0.7.1	259s
strobealign-0.13.0*	150-160s

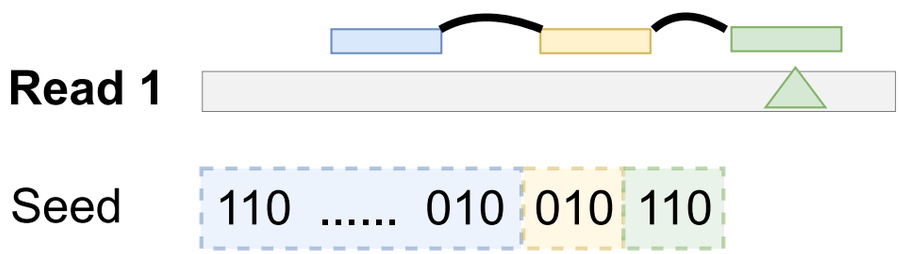
Multi-context seeds (MCS) improves accuracy

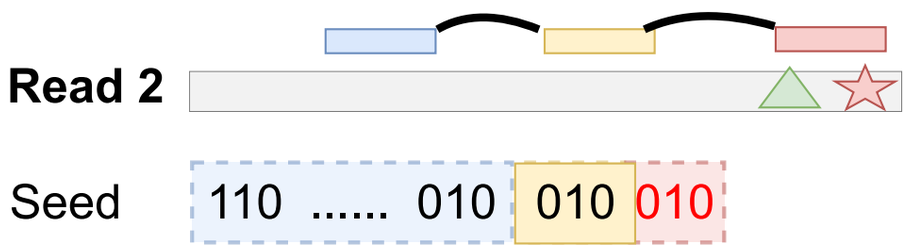
Strobe information lost!

- 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



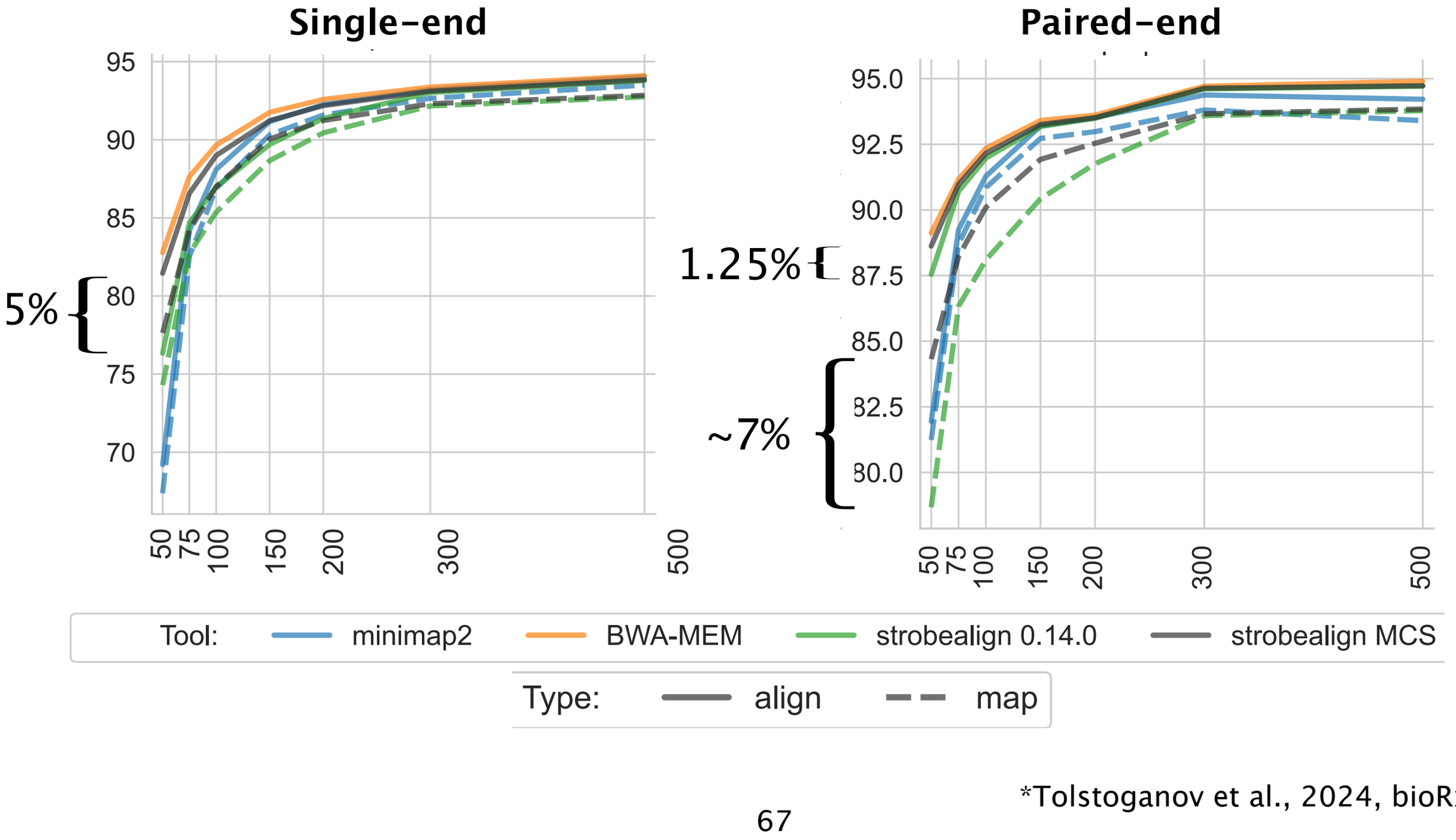






*Tolstoganov et al., 2024, bioRxiv

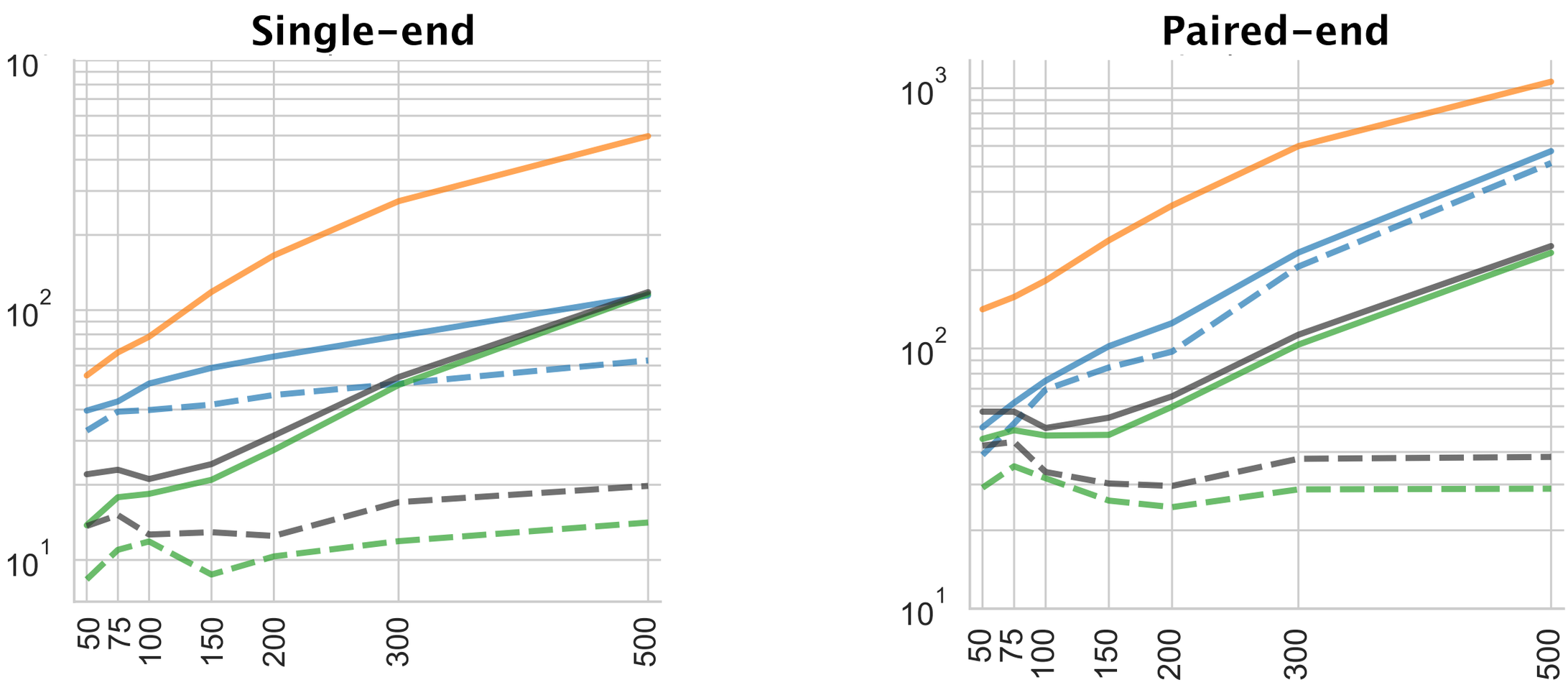
Mapping accuracy
(simulated Illumina reads to CHM13)



*Tolstoganov et al., 2024, bioRxiv

Mapping time
(simulated Illumina reads to CHM13)

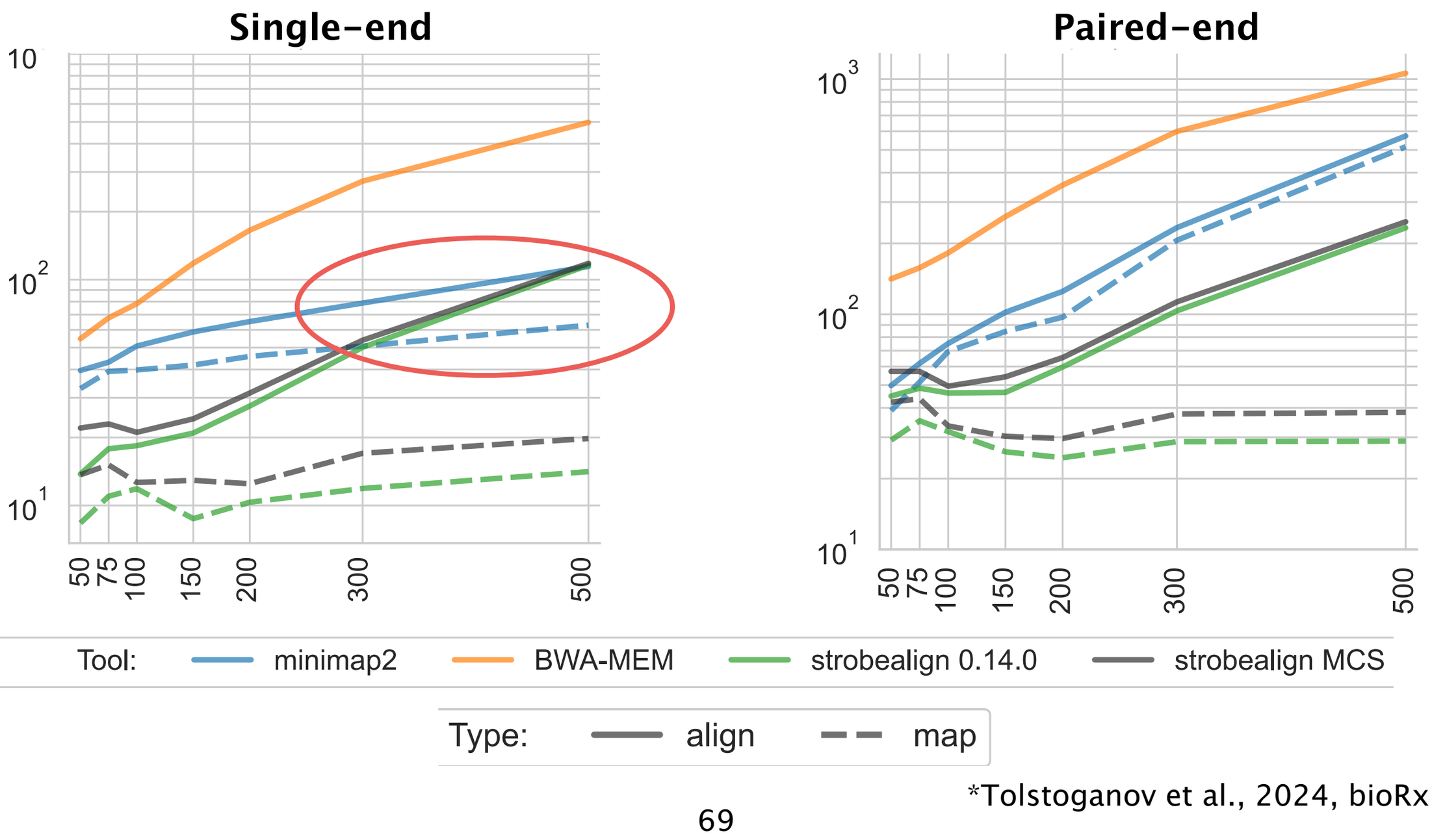
- strobealign-MCS is a bit slower than v0.14.0 (more seeds)
- We believe we can optimise aemb a bit



*Tolstoganov et al., 2024, bioRxiv

Mapping time
(simulated Illumina reads to CHM13)

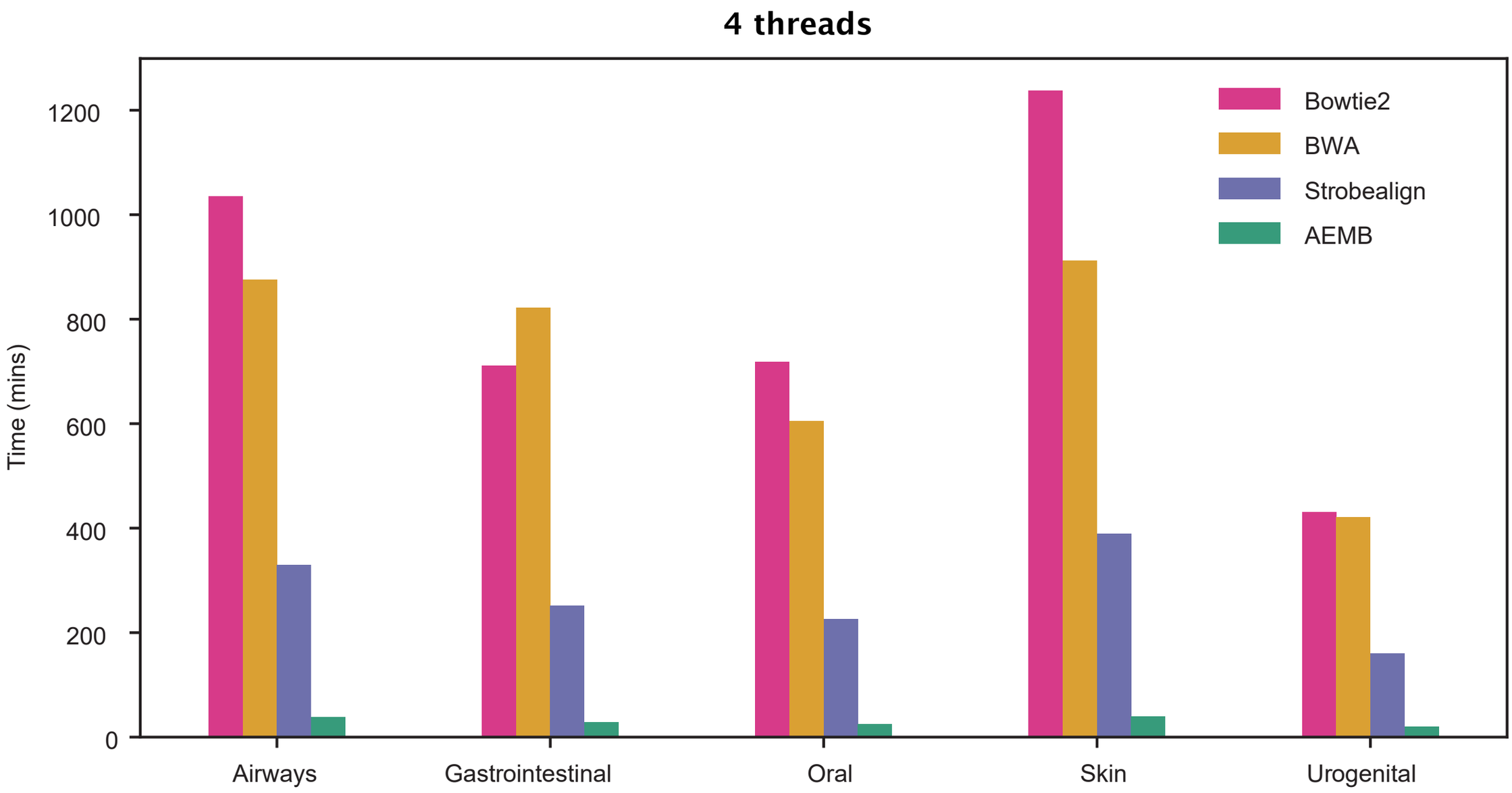
- strobealign-MCS is a bit slower than v0.14.0 (more seeds)
- We believe we can optimise aemb a bit
- Another conclusion: We need to do piece-wise (extension is a bottleneck)



*Tolstoganov et al., 2024, bioRxiv

strobealign new feature: --aemb

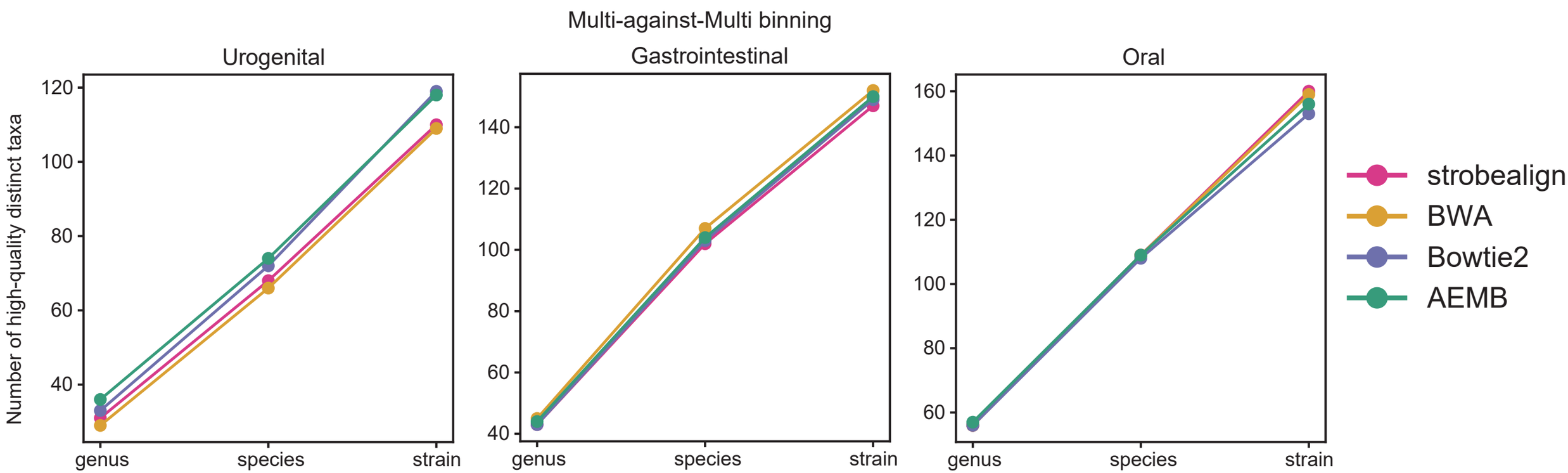
- **aemb**: “Abundance estimation metagenomic (contig) binning”
- Mapping-based coverage computation
- Multi-against-multi binning (Mattock and Watson, 2023)



*Pan et al., forthcoming study

strobealign new feature: --aemb

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



*Pan et al., forthcoming study

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

Funding

Swedish Research Council

NATIONAL BIOINFORMATICS INFRASTRUCTURE SWEDEN

LongTREC

Funded by the European Union

@ksahlin

@ksahlin.bsky.social

@krsahlin

Appendix

E-hits

x_i : count seed i
$p_i = x_i/N$
N : total seeds, M : distinct seeds

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

Sahlin, 2021, Genome Research

Conclusions Moein et al?

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

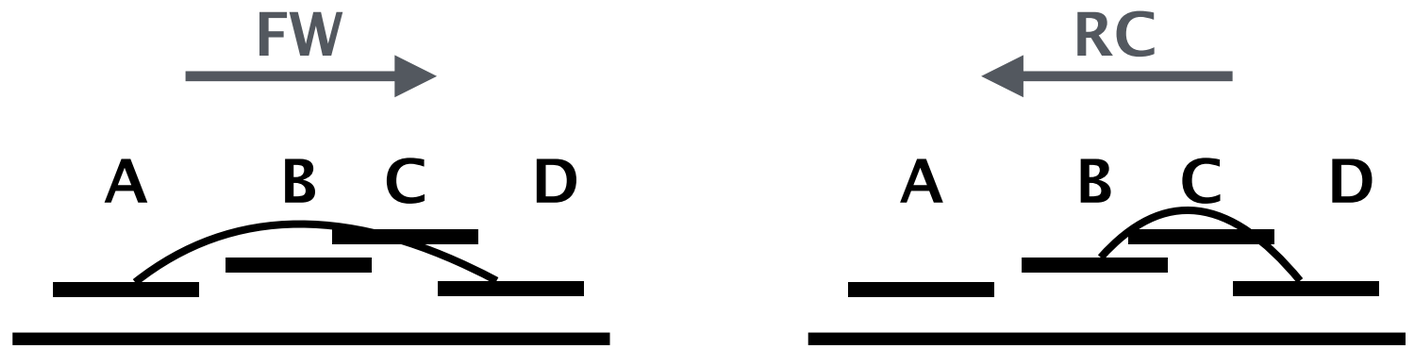
XOR with argmin: selects $s_2 = s_1$ if present

XOR with argmax – variations are selected for

Karami et al. 2024, Bioinformatics

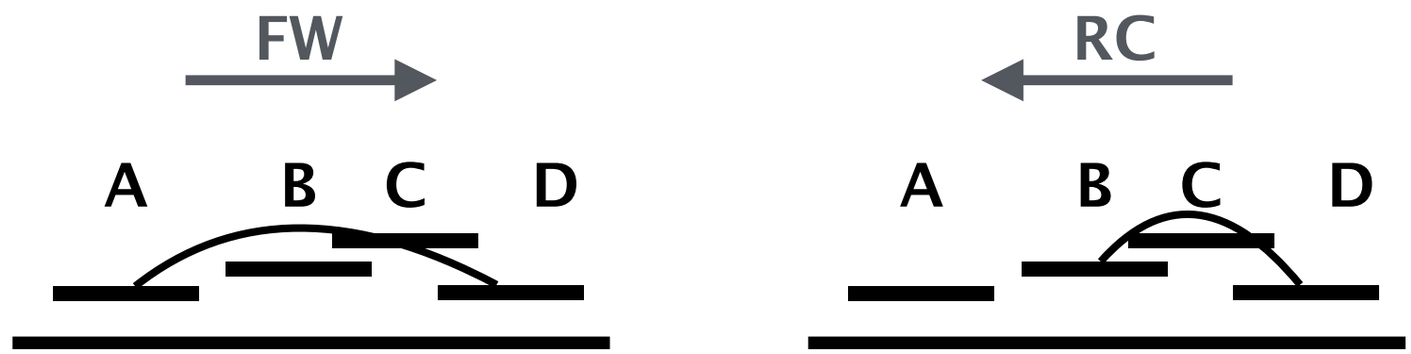
Strobemers: Limitations

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** 😕



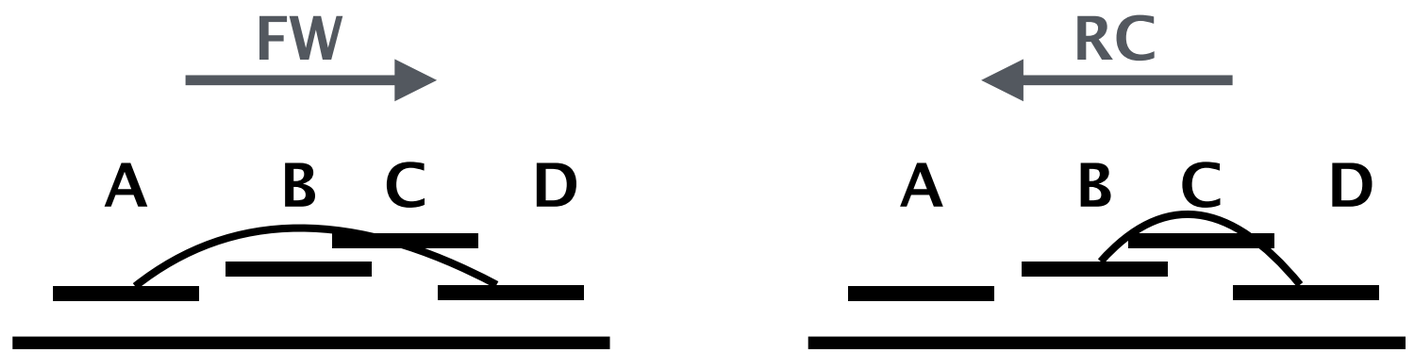
Remember: $f(X, Y)$ computes the hash value of the seed

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

Full canonicity

Strobemers: Limitations

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

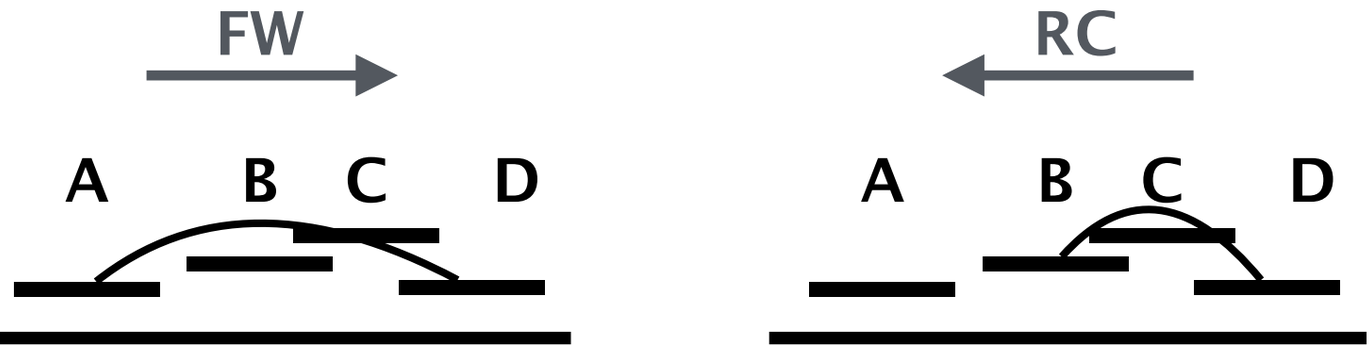


Remember: $f(X, Y)$ computes the hash value of the seed

	$\min\{f(A, D), f(D, B)\}$		Full canonicity
Symm.	$f(A, D) = f(D, A)$	Ex: A + B	'Some canonicity'(😛)
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

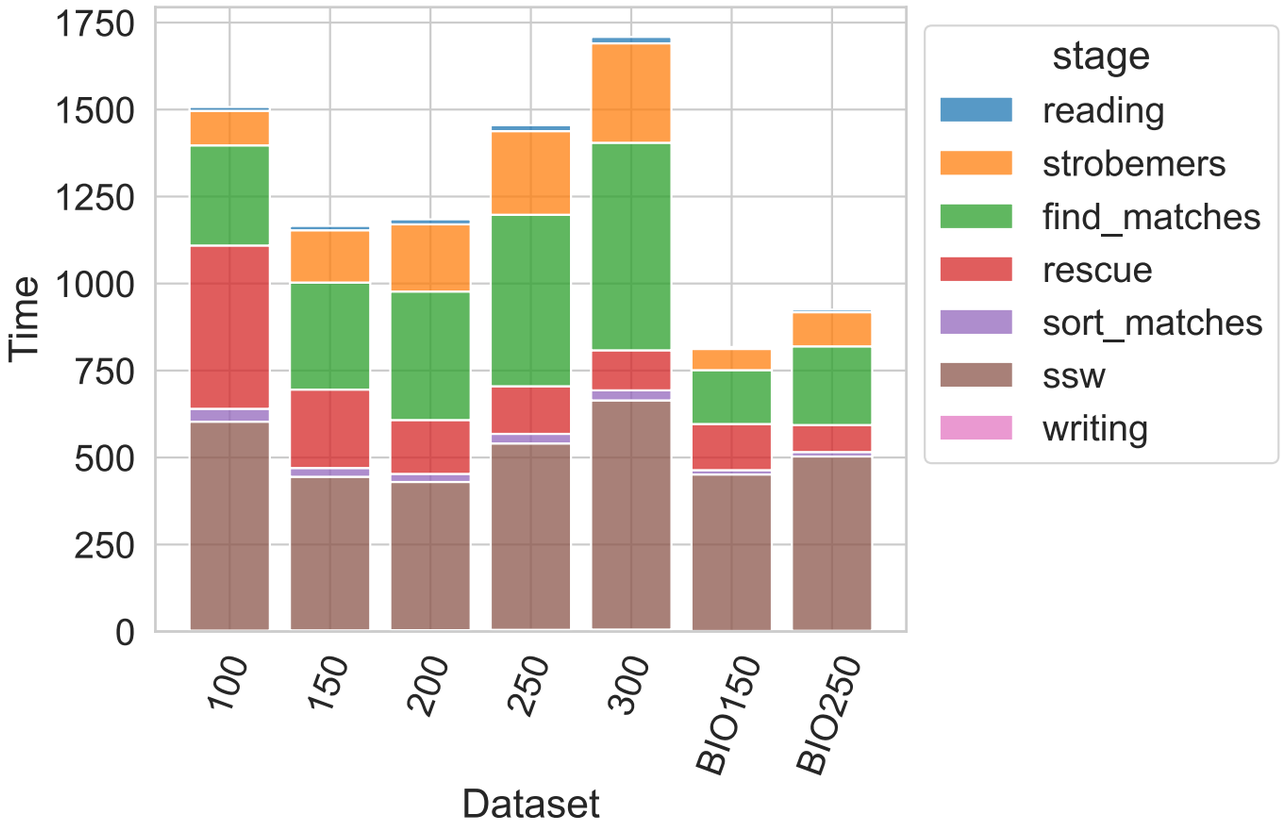
	$\min\{f(A, D), f(D, B)\}$		Full canonicity
Symm.	$f(A, D) = f(D, A)$	Ex: A + B	'Some canonicity' (😛)
Asymm.	$f(A, D) \neq f(D, A)$	Ex: 2A – B	No canonicity

Indexing: one pass (FW)
Querying: both FW and RC

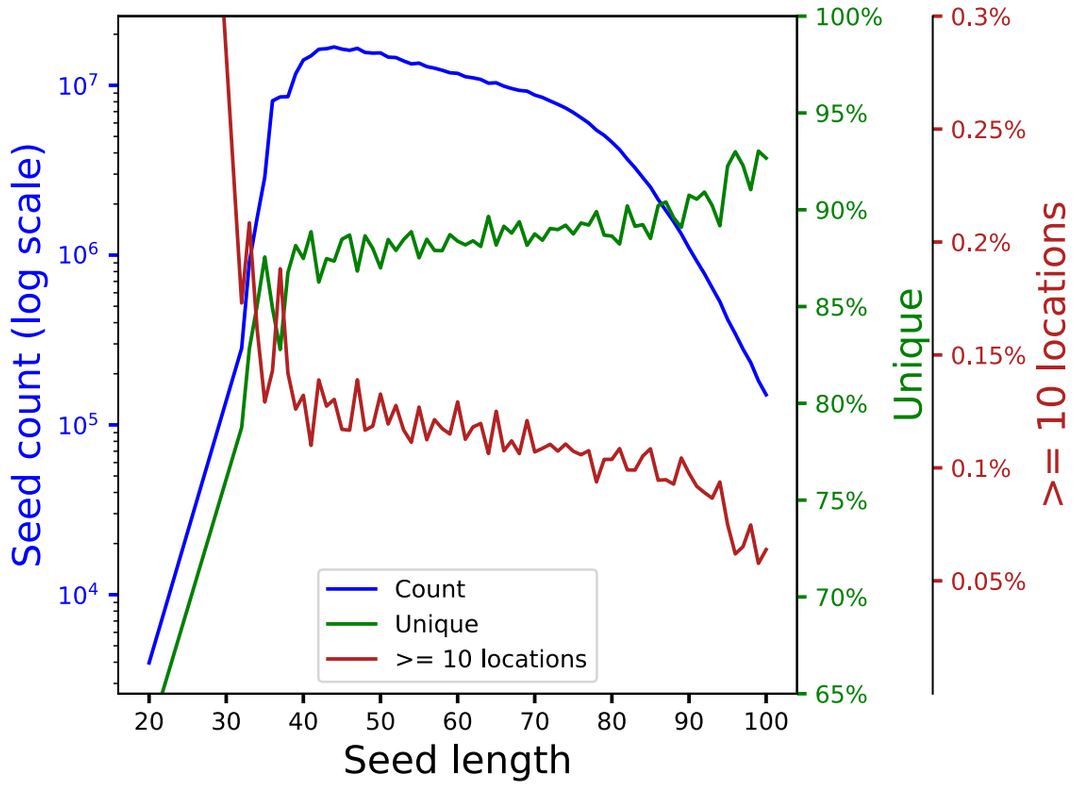
Symmetric f speeds up indexing, but at some cost:
inversions together with masking repeat masking seeds

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 i
$p_i = x_i/N$
N : total seeds, M : distinct seeds

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