Distributed String Mining for High-Throughput Sequencing Data

Niko Välimäki Simon J. Puglisi

Department of Computer Science
University of Helsinki
nvalimak@cs.helsinki.fi

String Mining

$$\mathcal{T}^+ = \{ \texttt{I am positive}, \texttt{I am also positive}, \texttt{I am also positive}\}$$

$$\mathcal{T}^- = \{ \texttt{I am negative}, \texttt{I am also negative}, \texttt{I am not negative}\}$$

Extract *emerging substrings* that discriminate the given datasets.

String Mining

$$\mathcal{T}^+ = \{ \texttt{I am positive}, \texttt{I am also positive}, \texttt{I am also positive}\}$$

$$\mathcal{T}^- = \{ \texttt{I am negative}, \texttt{I am also negative}, \texttt{I am not negative}\}$$

Extract *emerging substrings* that discriminate the given datasets.

String Mining

$$\mathcal{T}^+ = \{ \begin{array}{l} \texttt{I am positive}, \\ \texttt{I am also positive}, \\ \texttt{I am also positive} \} \end{array}$$

$$\mathcal{T}^- = \{ \begin{array}{l} \texttt{I am negative}, \\ \texttt{I am also negative}, \\ \texttt{I am not negative} \} \end{array}$$

Extract *emerging substrings* that discriminate the given datasets.

String Mining under Frequency Constrains

INPUT

- Sets $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_R$ of total length $n = \sum \|\mathcal{T}_i\|$.
- Constraint $(f^i_{\min}, f^i_{\max})$ for each $\mathcal{T}_i$

OUTPUT

- All substrings P s.t.

  $f^i_{\min} \leq \text{freq}(P, \mathcal{T}_i) \leq f^i_{\max}$

  for all i.

Where $\text{freq}(P, \mathcal{T})$ is number of strings $T \in \mathcal{T}$ s.t. P occurs in T.

String Mining under Frequency Constrains

INPUT

- Sets $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_R$ of total length $n = \sum \|\mathcal{T}_i\|$.
- Constraint $(f^i_{\min}, f^i_{\max})$ for each $\mathcal{T}_i$

OUTPUT

- All substrings P s.t.

  $f^i_{\min} \leq \text{freq}(P, \mathcal{T}_i) \leq f^i_{\max}$

  for all i.

Remark

It is non-trivial to choose $(f^i_{\min}, f^i_{\max})$; use e.g. χ^2 test instead.

Motivation: String Algorithms


Optimal String Mining Under Frequency Constraints

Johannes Fischer, Volker Heun, and Stefan Kramer


An Efficient Algorithm for Mining String Databases Under Constraints

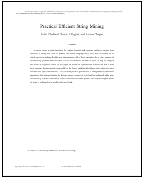
Sau Dan Lee and Luc De Raedt


Efficient String Mining under Constraints Via the Deferred Frequency Index

David Weese and Marcel H. Schulz


A space efficient solution to the frequent string mining problem for many databases

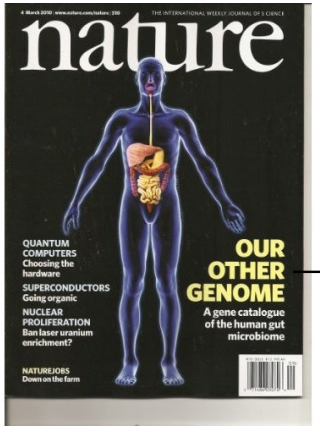
Adrian Kügel · Enno Ohlebusch


Practical Efficient String Mining

Jasbir Dhaliwal, Simon J. Puglisi, and Andrew Turpin

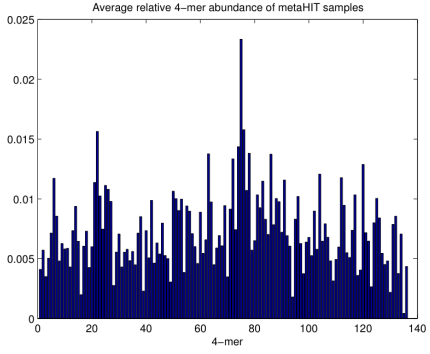
. . .

Motivation: Sequence Analysis

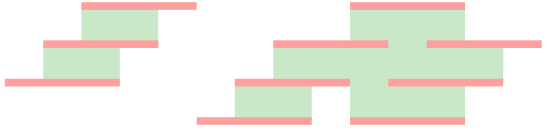


Human gut metagenome, 124 samples

50 million reads per sample



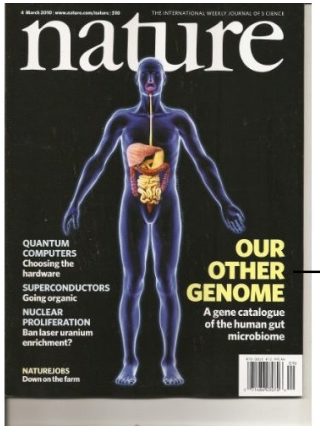
k-mers + prob. modeling



assembly + mapping

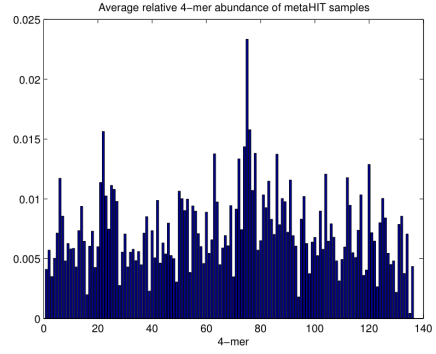
Mapping requires reference genomes.

Motivation: Sequence Analysis

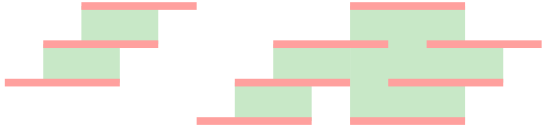


Human gut metagenome, 124 samples

50 million reads per sample



k-mers + prob. modeling



assembly + mapping

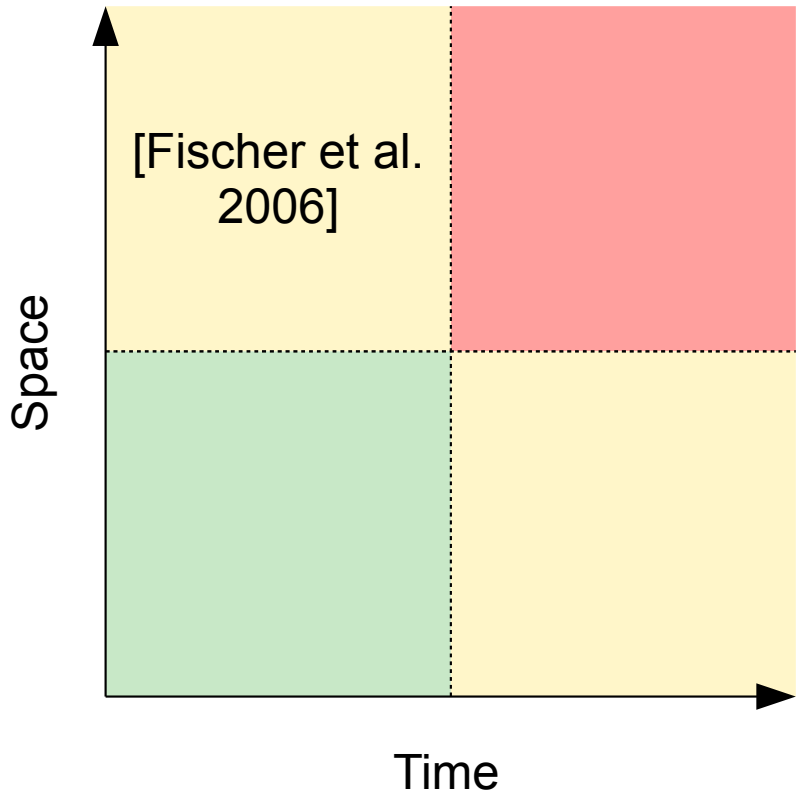
Replace k-mers with string mining? (both are *de novo*)

Earlier Work

[Fischer, Heun, Kramer 2006]

- Optimal $\mathcal{O}(n)$ time
- $\Theta(n \log n)$ bits

Requires 50-100 GB for human genome-scale inputs.

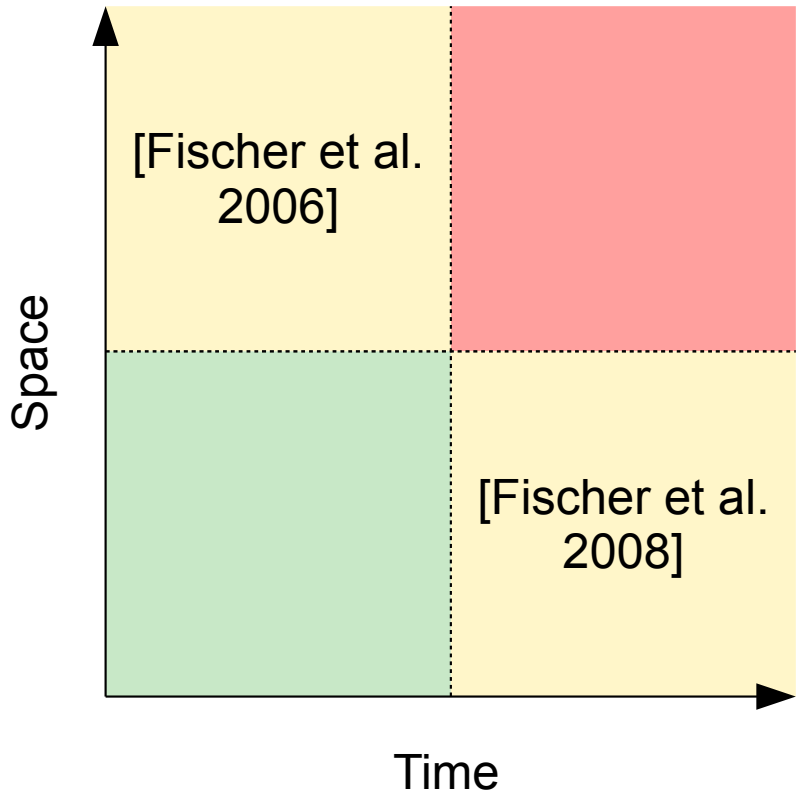


Earlier Work

[Fischer, Mäkinen, V 2008]

- $\mathcal{O}(n \log n)$ time
- $\mathcal{O}(n \log \sigma)$ bits

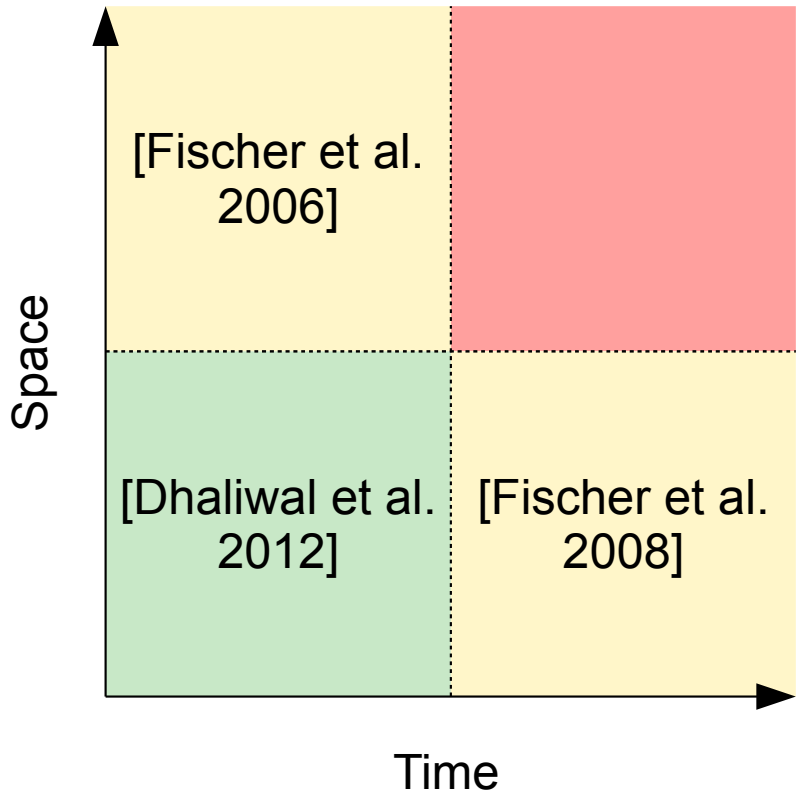
First to scale up to human genome sized inputs.



Earlier Work

[Dhaliwal, Puglisi, Turpin 2012]

- $\mathcal{O}(n \log^2 n)$ time
- $\mathcal{O}(n \log \sigma)$ bits



Optimal-Time Algorithm

Construct

1. *suffix array*,
2. *LCP* array + *RMQ*.

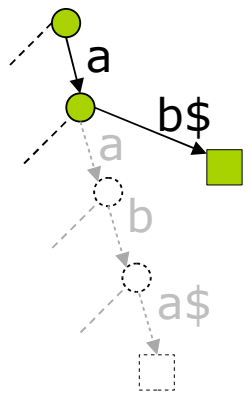
All in $\mathcal{O}(n)$ time.

Integration of

1. [Kasai et al. 2001] to visit all *branching substrings*,
2. [Hui 1992] to solve the *color set size problem*.

Both in $\mathcal{O}(n)$ time.

	1	2	3	4	5	6	7	8	9	10	11	12
SA:	5	12	18	23	4	22	8	9	1	10	2	6
LCP:	0	0	0	0	0	1	1	2	3	1	2	3
	$	$	$	$	a	a	a	a	a	a	a	a
					$	$	a	a	a	b	b	a
							a	b	b	$	a	a
							b	$	a		$	a
							$		$			b
												$



Optimal-Time Algorithm

Construct

1. *suffix array*,
2. *LCP* array + *RMQ*.

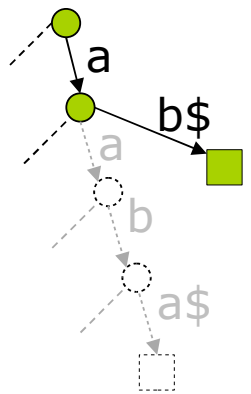
All in $\mathcal{O}(n)$ time.

Integration of

1. [Kasai et al. 2001] to visit all *branching substrings*,
2. [Hui 1992] to solve the *color set size problem*.

Both in $\mathcal{O}(n)$ time.

	1	2	3	4	5	6	7	8	9	10	11	12
SA:	5	12	18	23	4	22	8	9	1	10	2	6
LCP:	0	0	0	0	0	1	1	2	3	1	2	3
	$	$	$	$	a	a	a	a	a	a	a	a
					$	$	a	a	a	b	b	a
							a	b	b	$	a	a
							b	$	a		$	a
							$		$			b
												$



Summary of Earlier Work

State of the art methods require that:

1. the whole input fits in main memory (of one machine),
2. the computation is serial.

Our distributed algorithm solves both problems.

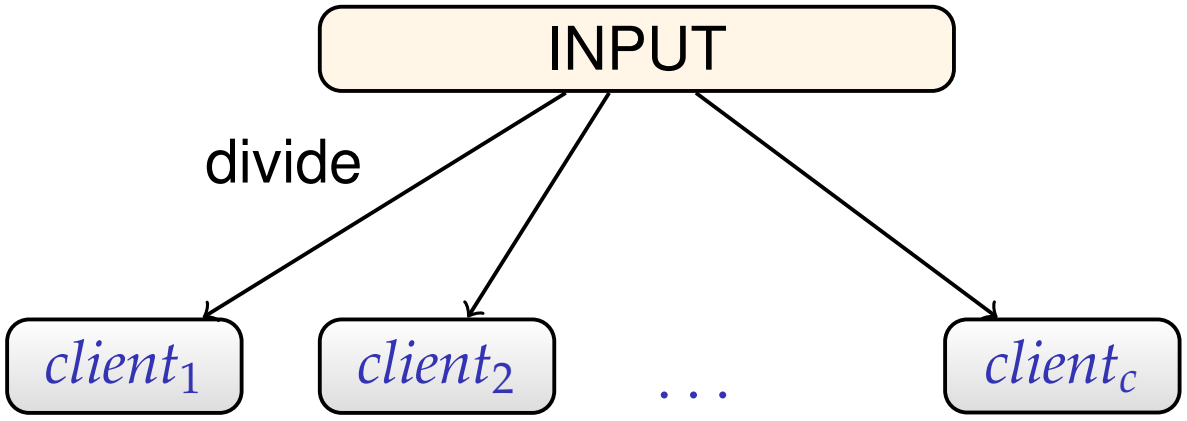
Summary of Earlier Work

State of the art methods require that:

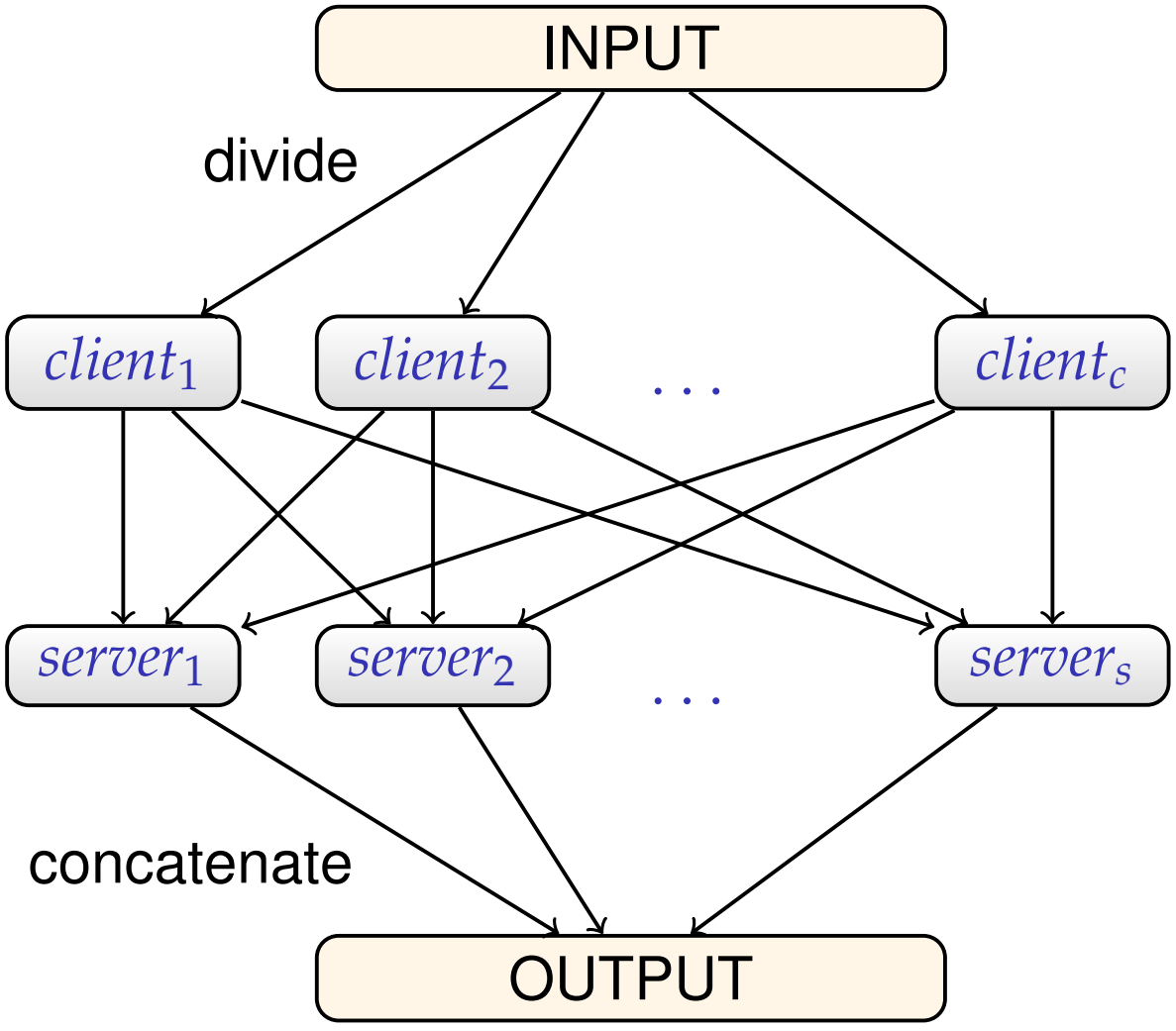
1. the whole input fits in main memory (of one machine),
2. the computation is serial.

Our distributed algorithm solves both problems.

Distributed String Mining



Distributed String Mining



Client Side Processing

1. Simulate a suffix tree traversal via *suffix array* & *LCP array*.
2. Compute frequencies and check against $f^i_{\min}$ and $f^i_{\max}$.

	Worst-case	Expected
Time	$\mathcal{O}\left(\max\{\ell, \frac{n}{c}\}\,\ell\right)$	$\mathcal{O}\left(\frac{n}{c}\log n\right)$
Space (in bits)	$\mathcal{O}\left(\max\{\ell, \frac{n}{c}\}\log n\right)$	

Space-efficiency: $\mathcal{O}\left(\frac{n}{c}\log\sigma\right)$ bits with $(\log n)$-factor slowdown.

Server Side Processing

1. Merge the (sorted) input from clients on the fly.
2. Output substrings that obey the constraints over all $\mathcal{T}^i$.

	Worst-case	Expected
Time	$\mathcal{O}\left(n \log n\right)$	$\mathcal{O}\left(\frac{n}{s} \log n\right)$
Space (in bits)	$\mathcal{O}\left(c\,\ell \log n\right)$	*negligible*
Transmitted bits	$\mathcal{O}\left(n \log^2 n\right)$	

Experimental Results

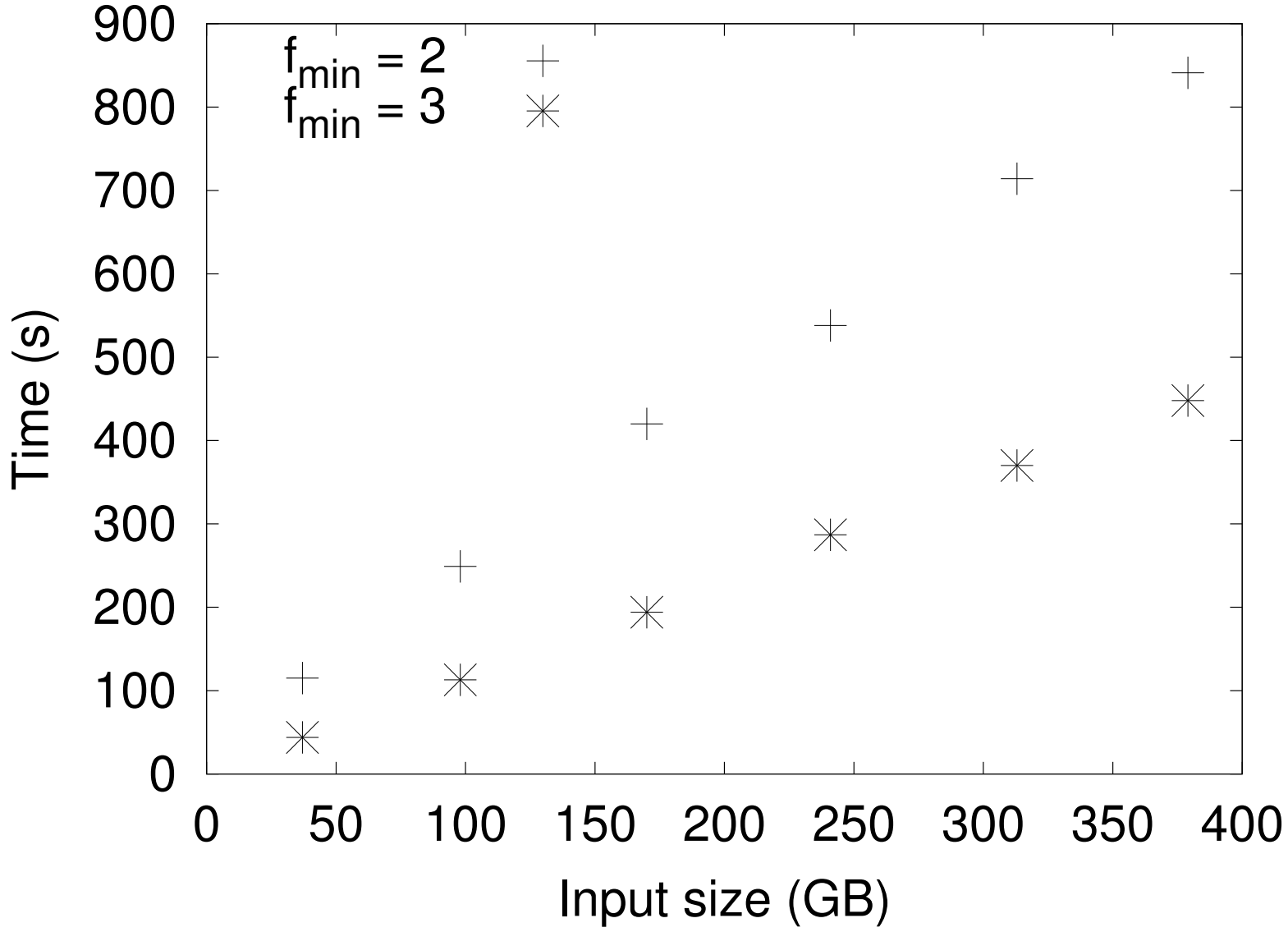
1. Human genome
   - Experiment given in [Fischer et al. 2008] [Dhaliwal et al. 2012].
2. Human gut metagenomics
   - 124 samples, 2.8 billion reads, 0.4 Tb.

Third experiment (in the paper) includes artificial data.

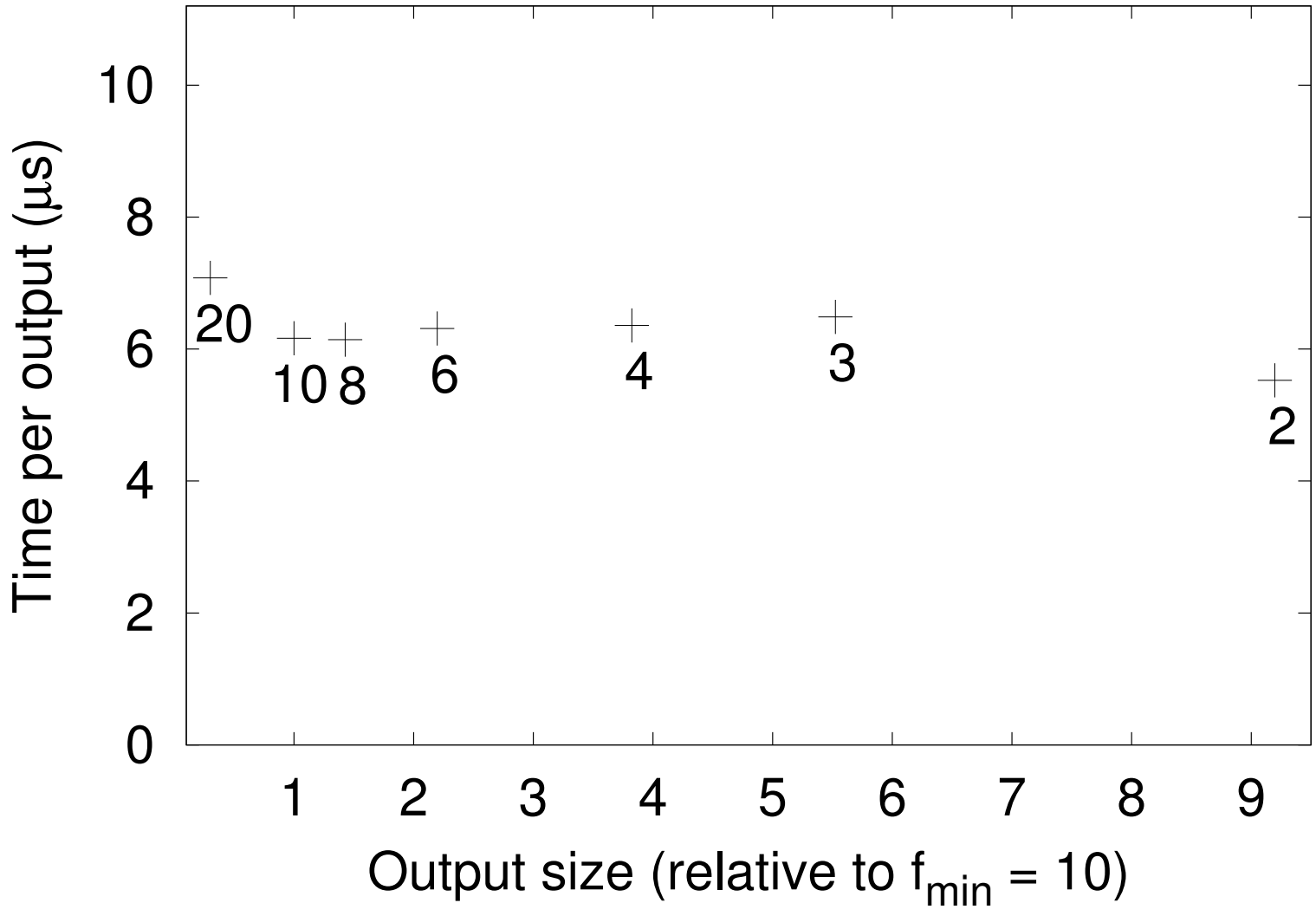
Human Genome-Scale Data

Method	Time	Memory
[Fischer et al. 2006]	1h	50.0 GB
[Fischer et al. 2008]	72h 12m	10.0 GB
[Dhaliwal et al. 2012]	3h 4m	17.7 GB
[Dhaliwal et al. 2012]	4h 27m	12.1 GB
[Dhaliwal et al. 2012]	5h 55m	9.3 GB
[Dhaliwal et al. 2012]	6h 4m	7.9 GB
Our	43m	4.9 GB

Human Gut Metagenomics



Human Gut Metagenomics



Summary

Earlier algorithms:

- require that the input fits main memory,
- scale up to gigabytes of input.

Distributed variant:

- improves time and space complexities,
- scales up to terabytes of input,
- $\approx$ \$500 to analyze 0.4TB at Amazon EC2.