

**University of Stuttgart**

Institute of Biomedical Genetics

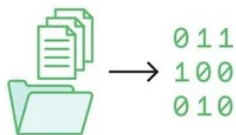
University of Stuttgart

# Efficient trace reconstruction in DNA storage systems using Bidirectional Beam Search

**Zhenhao Gu,<sup>1,2</sup> Hongyi Xin,<sup>3</sup> Puru Sharma,<sup>1</sup> Gary Yipeng Goh,<sup>1</sup>  
Limsoon Wong<sup>1,\*</sup> and Niranjan Nagarajan<sup>1,2,4,\*</sup>**

<sup>1</sup>Department of Computer Science, School of Computing, National University of Singapore, 117417, Singapore, <sup>2</sup>Genome Institute of Singapore, A\*STAR, 138672, Singapore, <sup>3</sup>Global Institute of Future Technology, Shanghai Jiao Tong University, 200240, Shanghai, China and <sup>4</sup>Yong Loo Lin School of Medicine, National University of Singapore, 117596, Singapore

# Introduction



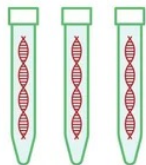
**1** For DNA data storage, files are first represented in binary code.



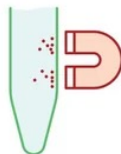
**2** Next, that binary code is converted into DNA sequences.



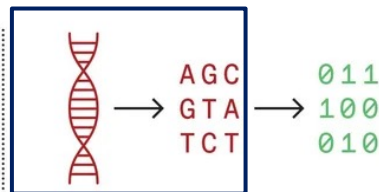
**3** Then DNA strands of those specific sequences are synthesized, using either today's chemical synthesis methods or more advanced enzymatic synthesis.



**4** The DNA is stored, perhaps in a test tube or a silica particle.



**5** When the data is needed, specific strands of DNA are extracted from storage, for example by using magnetic beads.



**6** The DNA strands are sequenced using one of several available methods, and then that sequence is converted back into binary code.

*Chris Phiplot: IEEE Spectrum*

# Motivation and intuition

Given: multiple noisy versions (called **traces**) of an original DNA sequence

Goal: recover the original string as accurately as possible.

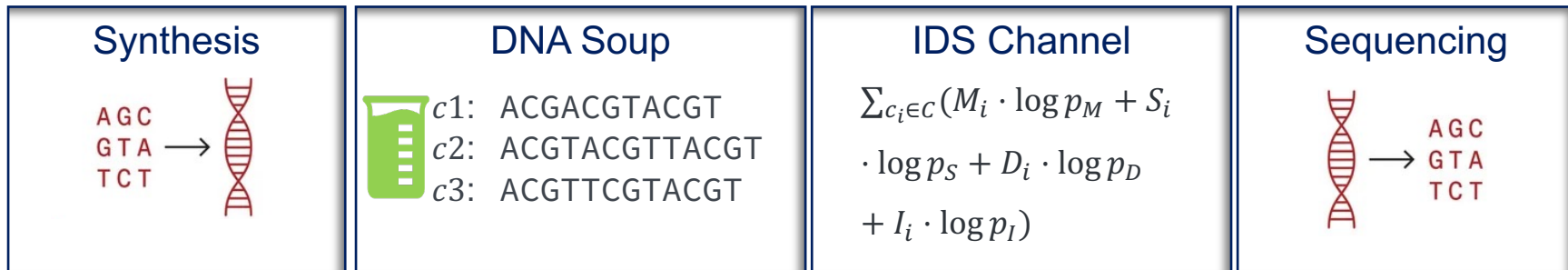
- Original seed/codeword: ACGTACGTACGT
- Synthesized into DNA strands and stored
- Later retrieved via sequencing (e.g., Nanopore), producing noisy traces:

$t_1$ :	ACGACGTACGT	(deletion)
$t_2$ :	ACGTACGTTACGT	(insertion)
$t_3$ :	ACGTTCGTACGT	(substitution)

# Trace reconstruction vs sequencing

Feature	Trace reconstruction	Sequencing
Goal	Recover known encoded sequence	Discover unknown genome
Receiver knows seed?	Partly (length, encoding, ECC)	No
Trace characteristics	Noisy versions of same seed	Short reads from unknown genome

# Integrating ECC and sequencing together



- Each trace  $c_i \in C$  is independent.
- Each trace  $c_i$  is a modified version of the seed string  $s \in \Sigma^L$ 
  - Insertion  $p_I$
  - Deletion  $p_D$
  - Substitution  $p_S$
  - Matching  $p_M$
- Objective:  
 $\arg \max (Pr[C | \hat{s}]).$

## State of the art

- CPL – conditional probability logic using probabilistic refinement
  - Trellis Bitwise Majority Alignment – IDS-channel based model
  - MUSCLE – multiple sequence alignment based method
  - ITR – iterative trace reconstruction or using consensus between trace pairs
- > **BBS – bidirectional beam search algorithm paired with k-th order Markov chain**

# CPL – conditional probability logic

Step 1: Align all traces to the first trace using edit distance

Step 2: Calculate probabilities for Markov chain using counts

$$Pr[x | y] = \frac{Count(s_{i-1} \rightarrow s_i)}{\sum Count(s' \rightarrow s_i)}$$

Step 3: Calculate the final trace using the most probable values in a matrix

$$\operatorname{argmax}(\log P(s_1) + \sum \log P(s_i | s_{i-1}))$$

Final trace is reconstructed using longest path in a graph constructed from characters

**Time complexity:**  $O(NL^2)$  for alignments +  $O(L|\Sigma|^2)$  for path finding

A G T C C A C T T T  
A G T C C G C T T T  
A G T T C G C T A T

	A	C	G	T
A	0	0.5	0.5	0
C	0	0.42	0.42	0.16
G	0	0.5	0	0.5
T	0.2	0.4	0	0.4

A G T C C G C T C T

# Trellis bitwise majority alignment

Step 1: Pick a predefined HMM based on error characteristics of Nanopore

Step 2: Build a Trellis graph for full cluster of  $N$  traces

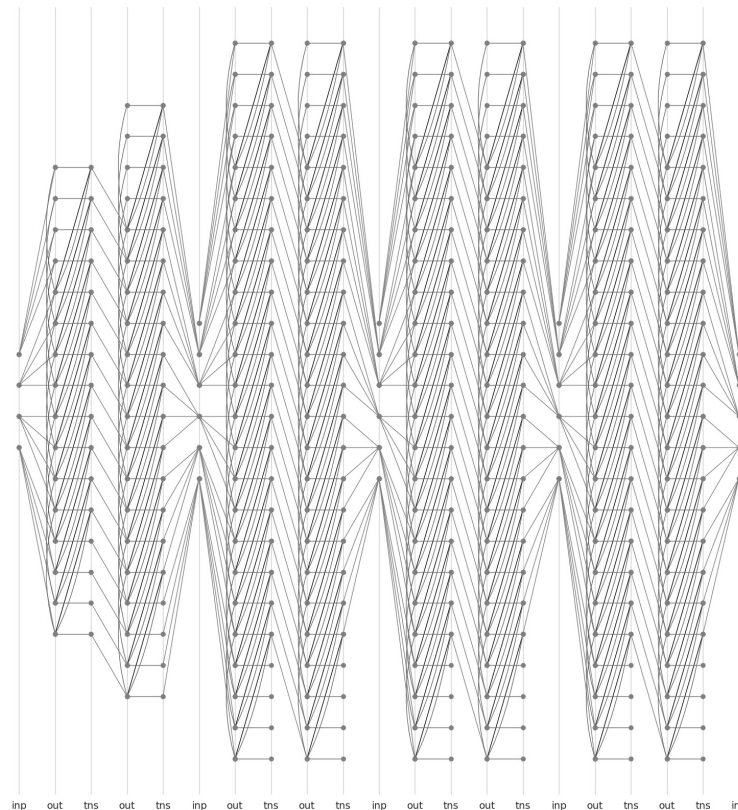
Step 3: Maximize the path along the Trellis with  $N$  traces

$$\hat{s} = \operatorname{argmax} \sum \log \Pr[c_i | s]$$

Time complexity:  $O(R \cdot N \cdot L \cdot T)$

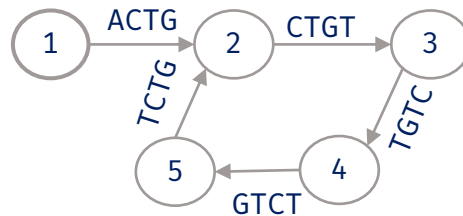
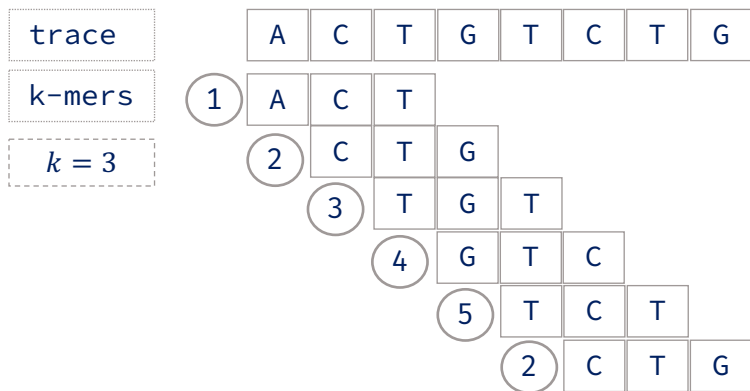
$R$ : refinement rounds,  $N$ : number of traces,  
 $L$ : seed length,  $T$ : trace length

Trellis for a 4x10 traces

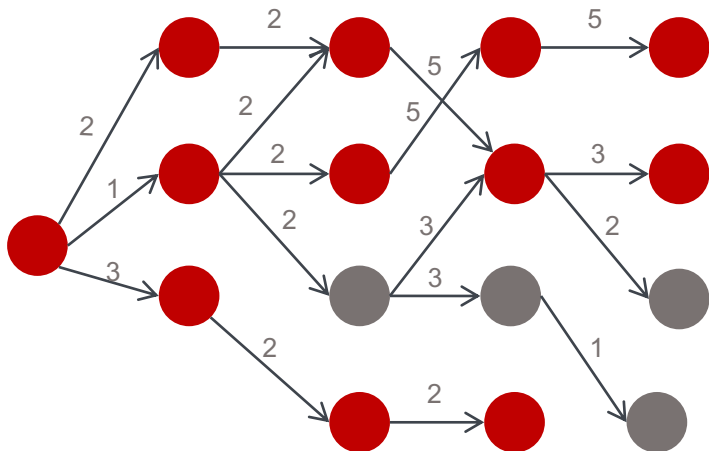


# Methods

- Count all the k-mers and k+1-mers of all the traces
- Based on the counts build a probabilistic model/k-th order Markov chain
- Build a de Bruijn graph out of those k-mers
- Execute bidirectional beamsearch – forward beamsearch and backward beamsearch
- Pick best of two aggregate probability profiles



## Beam search



### Drawbacks of Beamsearch:

1.  $B$  is not “adaptable” and thus we might end up with “bad paths”
2. Sometimes it is tough to choose a criterion as pruning might hurt the outcome

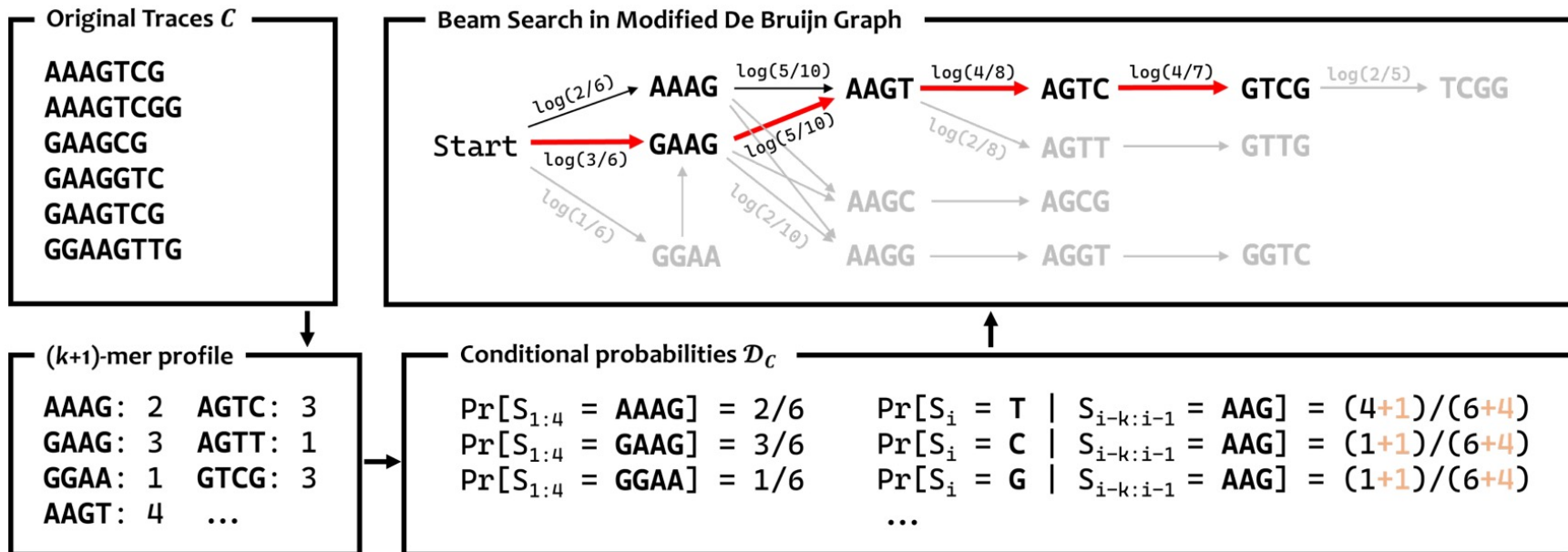
- Early pruning based on:
    - Beam width – B for priority queue
    - Criterion for accumulating best paths
    - BFS search
  - E.g.:
    - B=3
    - $\operatorname{argmax}(\sum w_i),$ 
      - (2,1,3)
      - (4,3,5)
      - (9,8,7)
      - (8,10,7)
- Advantages of beam search

  1. Aggressive pruning of paths with low scores, leading to reasonable complexity even for huge datasets
  2. With smart pruning, beam search might provide better results than exhaustive search

### Advantages of the Beamsearch:

1. Aggressive pruning allows a reasonable time and memory complexity and a runtime in huge datasets.
2. With smart pruning criterion it might prove to be very accurate.

# Bidirectional beamsearch and k-th order Markov chain



## Experimental setup

Dataset	Bar-Lev et al. [2]	Srinivasavaradhan et al. [36]	Chandak et al. [7]	
#Clusters	10000	9984	1466	<i>Sets of codewords each containing many traces</i>
Synthesis tech <sup>1</sup>	TB	TB	CA	<i>Twist Bioscience, CA = CustomArray</i>
Sequencing tech	MinION	MinION	MinION	<i>Nanopore MinION</i>
Clustering alg.	Bar-Lev et al. [2]	Rashtchian et al. [30]	Perfect	<i>How the sets of traces were clustered</i>
$L$	140	110	108	<i>Length of seeds/codewords</i>
Coverage	21.37	27.01	114.29	<i>Coverage of the traces</i>
$p_D$	1.17%	1.86%	5.09%	<i>Deletions</i>
$p_I$	1.52%	2.14%	4.56%	<i>Insertions</i>
$p_S$	1.65%	1.77%	3.91%	<i>Substitutions</i>
Error rate	4.34%	5.77%	13.56%	<i>Total error rate</i>

# Results

Dataset	Tools	Success rate	Avg. edit distance	Avg. Hamming distance	Running time (s)
Bar-Lev et al. [2]	MUSCLE	96.25%	0.258	1.382	1475.55
	Trellis BMA	92.41%	0.363	1.635	19759.85
	ITR	97.42%	0.221	0.975	12462.04
	CPL	<u>98.39%</u>	<b>0.201</b>	<u>0.374</u>	<u>495.22</u>
	BBS	<b>98.80%</b>	<u>0.206</u>	<b>0.346</b>	<b>23.84</b>
Srinivasavaradhan et al. [36]	MUSCLE	84.70%	0.259	6.032	1741.65
	Trellis BMA	69.57%	0.908	6.003	17859.17
	ITR	87.58%	0.232	4.797	7351.52
	CPL	<b>94.93%</b>	<b>0.150</b>	<b>1.286</b>	<u>361.05</u>
	BBS	<u>94.77%</u>	<u>0.168</u>	<u>1.616</u>	<b>20.01</b>
Chandak et al. [7]	MUSCLE	76.94%	0.366	8.821	7082.75
	Trellis BMA	52.32%	1.608	13.094	10744.21
	ITR	64.12%	0.666	12.231	1614.64
	CPL	<u>90.93%</u>	<b>0.205</b>	<b>2.299</b>	<u>131.60</u>
	BBS	<b>91.34%</b>	<u>0.283</u>	<u>2.738</u>	<b>8.52</b>

# Results

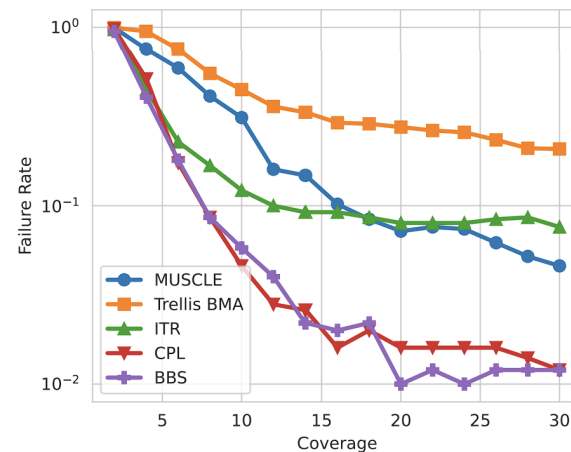
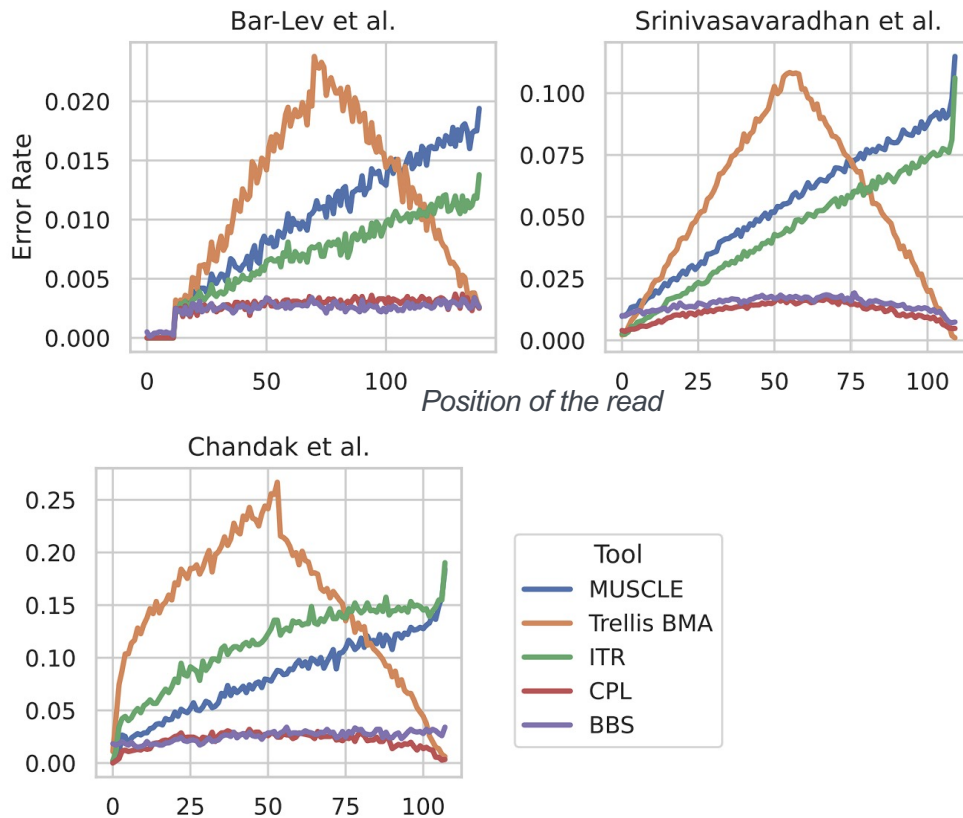


Fig. 4: Performance of the algorithms at small coverages in the dataset from Srinivasavaradhan et al.