A tale of optimizing the space usage of de Bruijn graphs

Rayan Chikhi

Institut Pasteur & CNRS

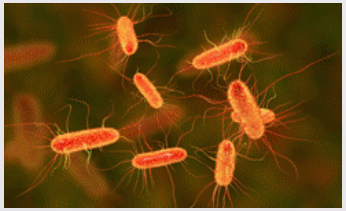
CiE 2021

Hello

- Bioinformatician / Computer scientist
- Head of the *Algorithms for biological sequences* group at Institut Pasteur

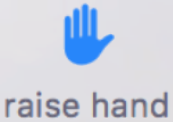
Research:

- Algorithms & data structures applied to DNA sequences
- Bioinformatics software
- Genome analysis





Please feel free to interact during this Zoom talk



Slides available

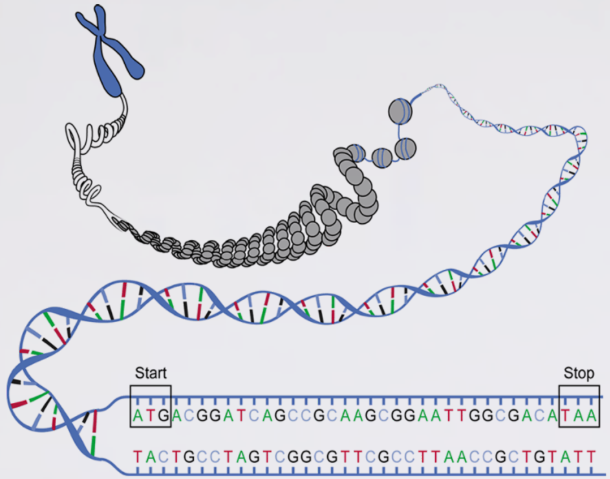
Slides of this talk are on my web page.

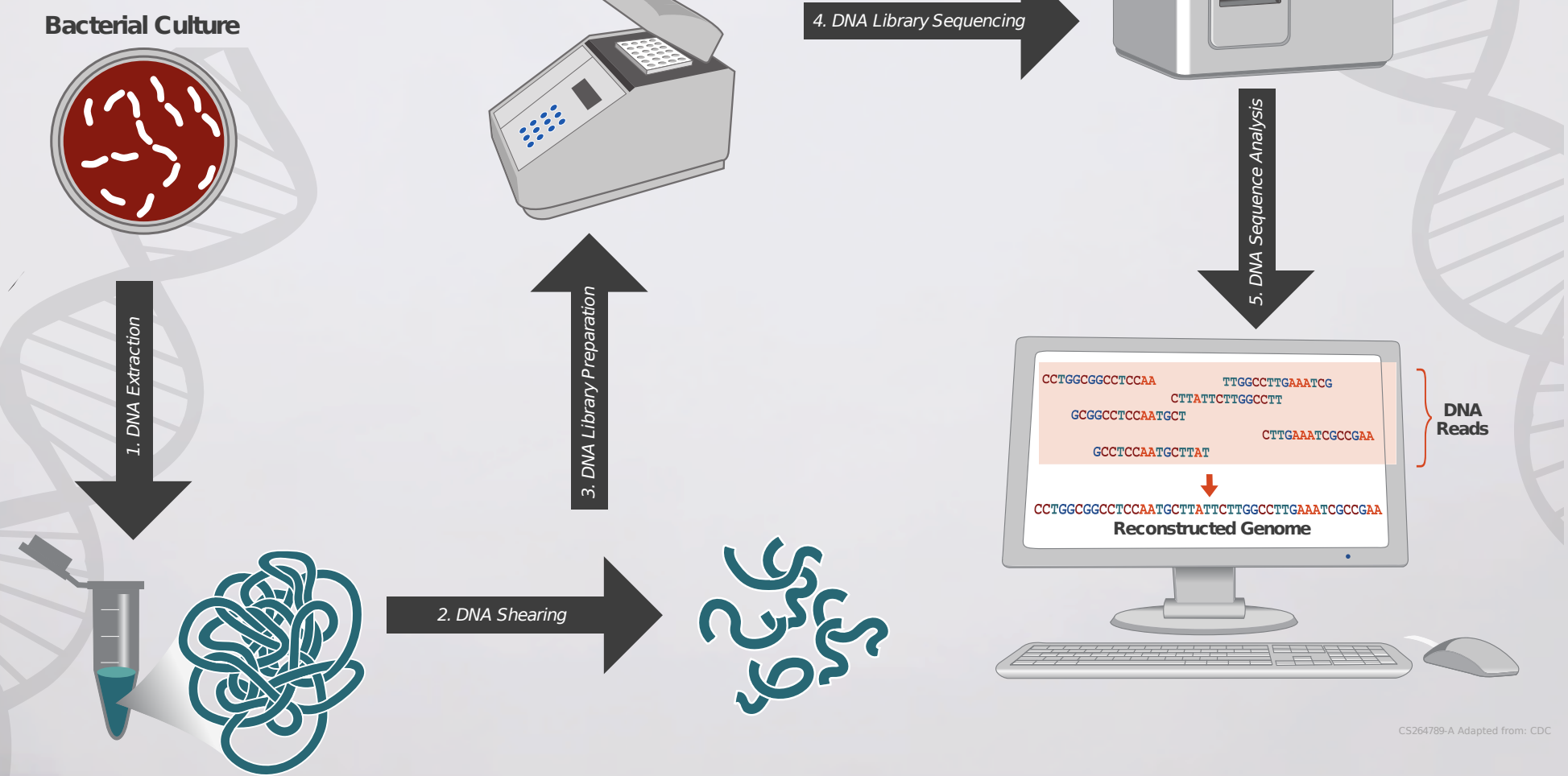
`http://rayan.chikhi.name`

Companion article in CiE proceedings & on Twitter (no account needed)

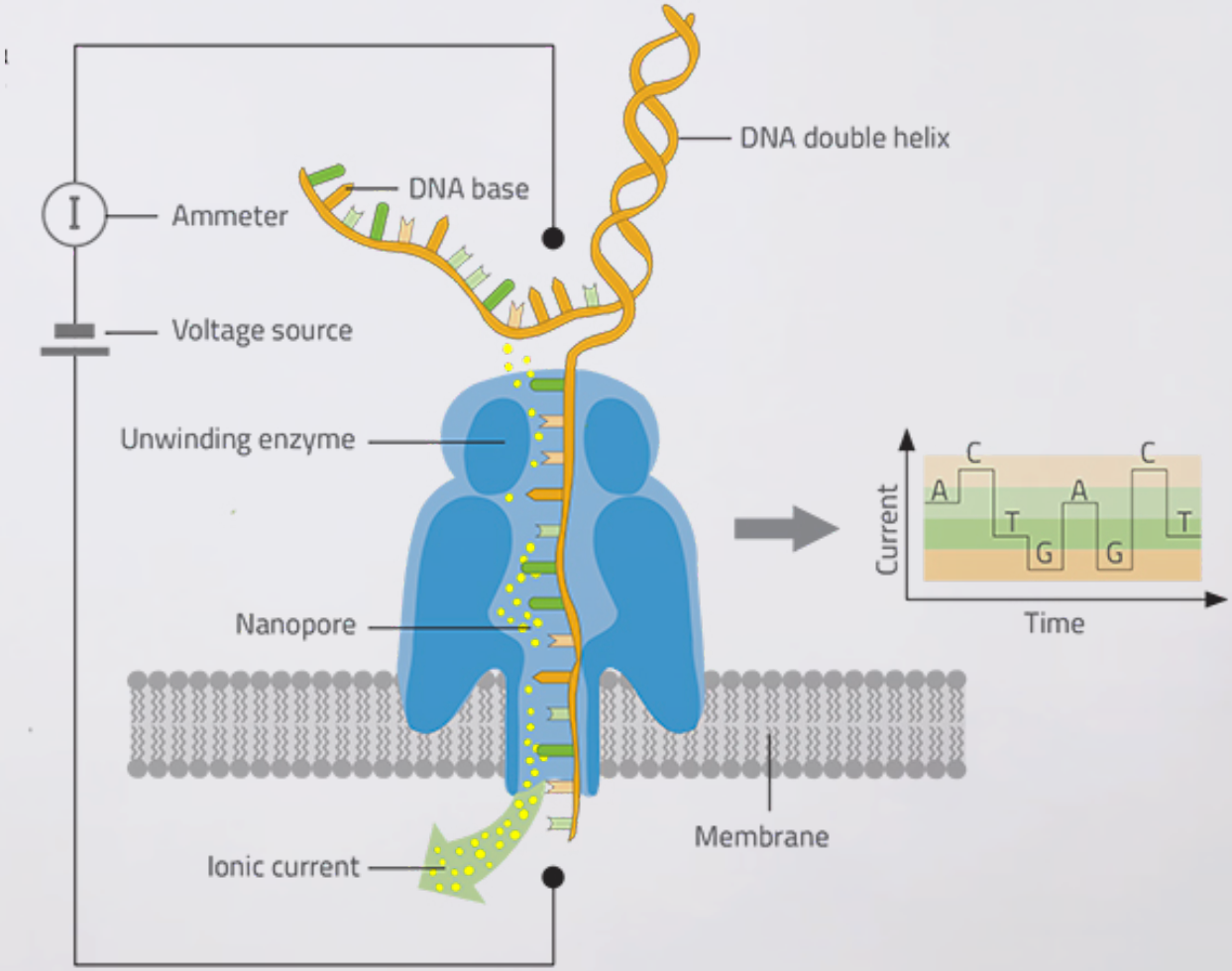
@RayanChikhi

DNA Sequencing





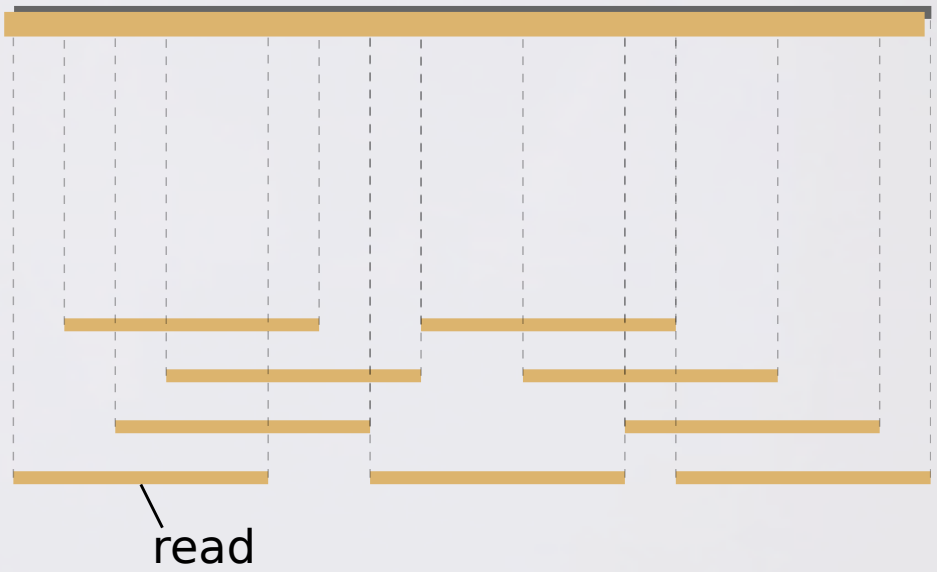
Long-read, portable DNA sequencing (Oxford Nanopore)



Genome assembly

genome
(unknown)

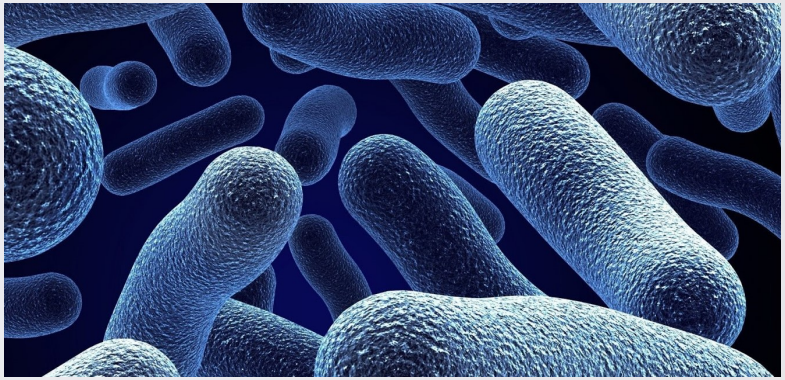
sequenced reads:
overlapping sub-sequences, covering the genome redundantly



assembly
hypothesis of the genome

The many applications of assembly

- **Reconstruct** genomes
- transcriptomes
- metagenomes
- genes
- Phylogeny of species
- Evolution of genes
- Find novel **insertions**
- **SNPs** in non-model organisms
- Structural variants
- Pangenomics
- . . .



40 years of whole-genome assembly

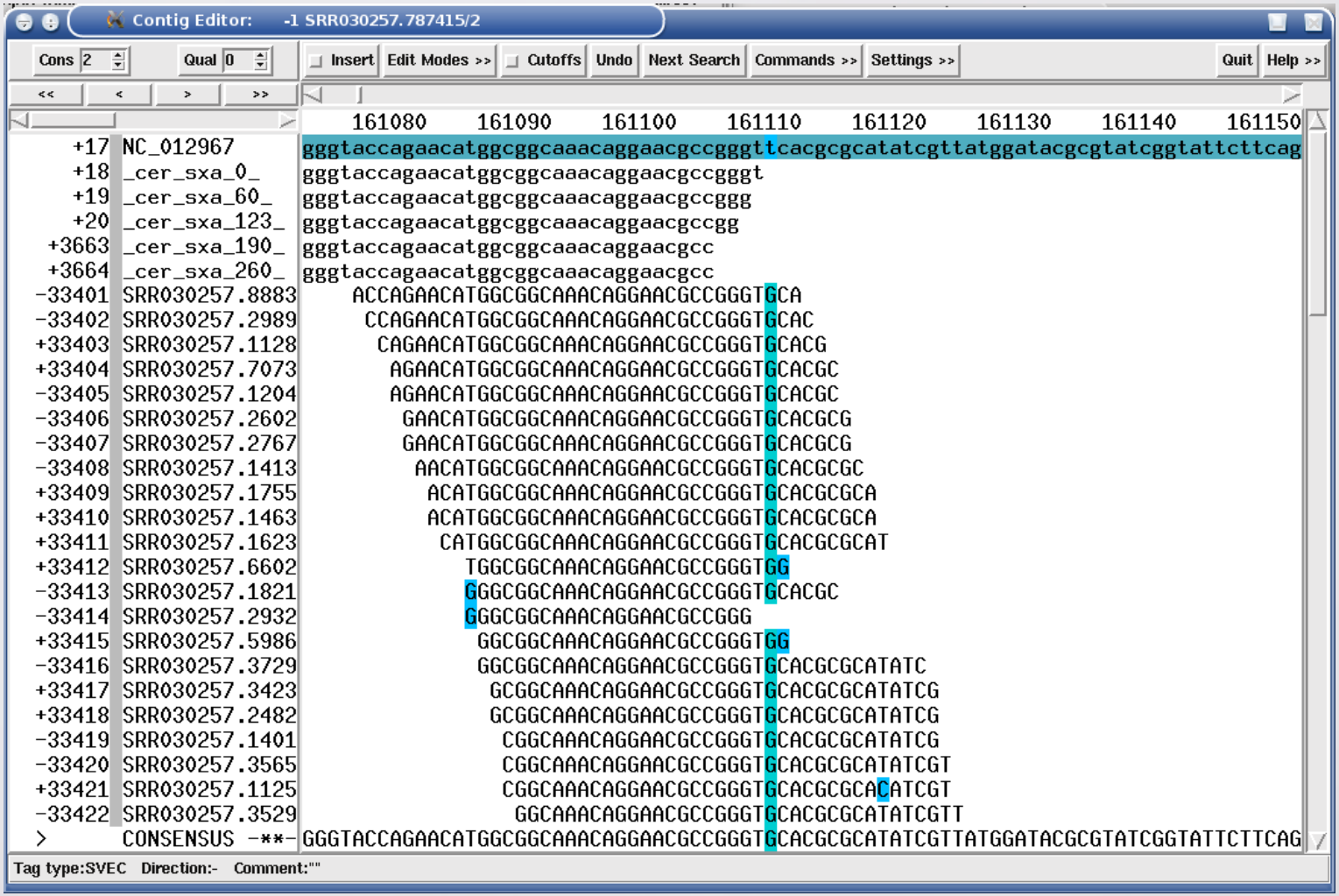
(Staden 1979) *"With modern fast sequencing techniques and suitable computer programs it is now possible to sequence whole genomes without the need of restriction maps."*

(A. Phillippy, RECOMB-Seq'19)

- Human Genome Project: completed in **2003**
- Truly completed in **2021** (T2T)
- First $1,000 genome in **2014**

A short algorithmic history of genome assembly

A short algorithmic history of genome assembly



Screenshot: MIRA

A short algorithmic history of genome assembly

1. Shortest Common Superstring
   - Find a min-length string containing all input strings (NP-hard)
2. Greedy algorithms

A short algorithmic history of genome assembly

1. Shortest Common Superstring
   - Find a min-length string containing all input strings (NP-hard)
2. Greedy algorithms
3. String graphs and de Bruijn graphs, both introduced at the same conference (DIMACS) in 1994

A History of DNA Sequence Assembly, G. Myers, 2016

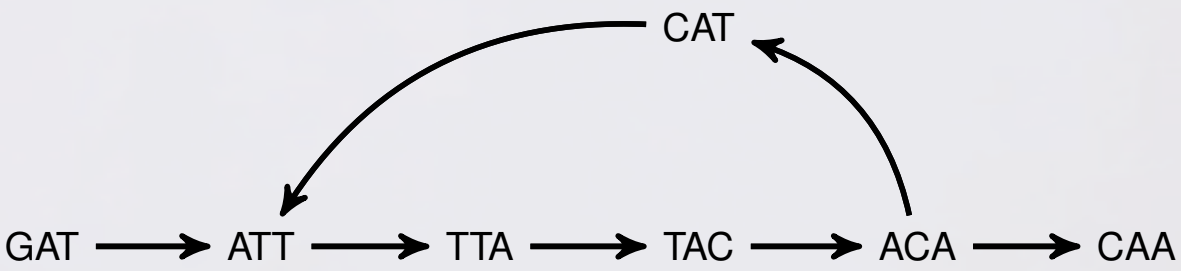
A short algorithmic history of genome assembly

1. Shortest Common Superstring
   - Find a min-length string containing all input strings (NP-hard)
2. Greedy algorithms
3. String graphs and de Bruijn graphs, both introduced at the same conference (DIMACS) in 1994

A History of DNA Sequence Assembly, G. Myers, 2016

Modern genome assembly: graphs

1. Construct a graph
2. Nodes are sequences
3. Edges are overlaps



Theory says.. [Nagarajan 09]

4. Return a path of *minimal length* that traverses **each node at least once**.

de Bruijn graphs

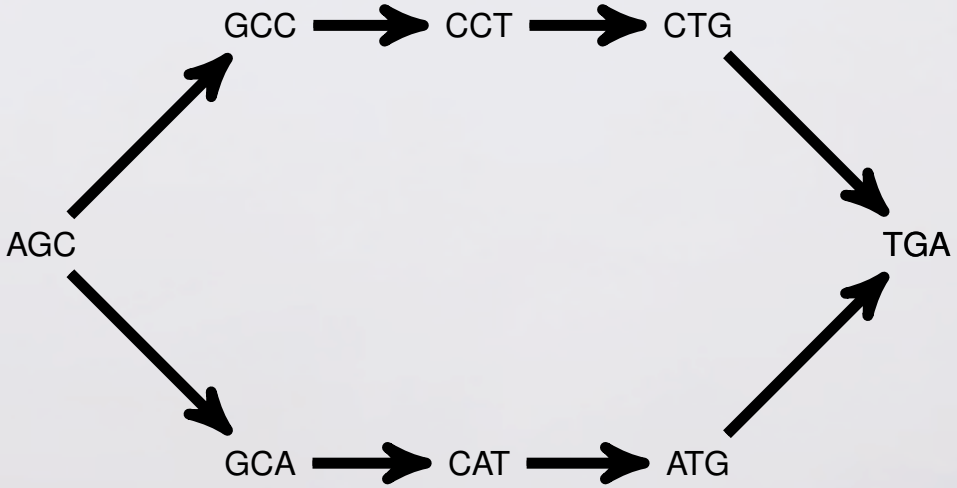
A **de Bruijn** graph for a fixed integer k:

1. **Nodes** = all k-*mers* (substrings of length k) in the reads
2. **Edges** = all exact overlaps of length exactly $(k-1)$

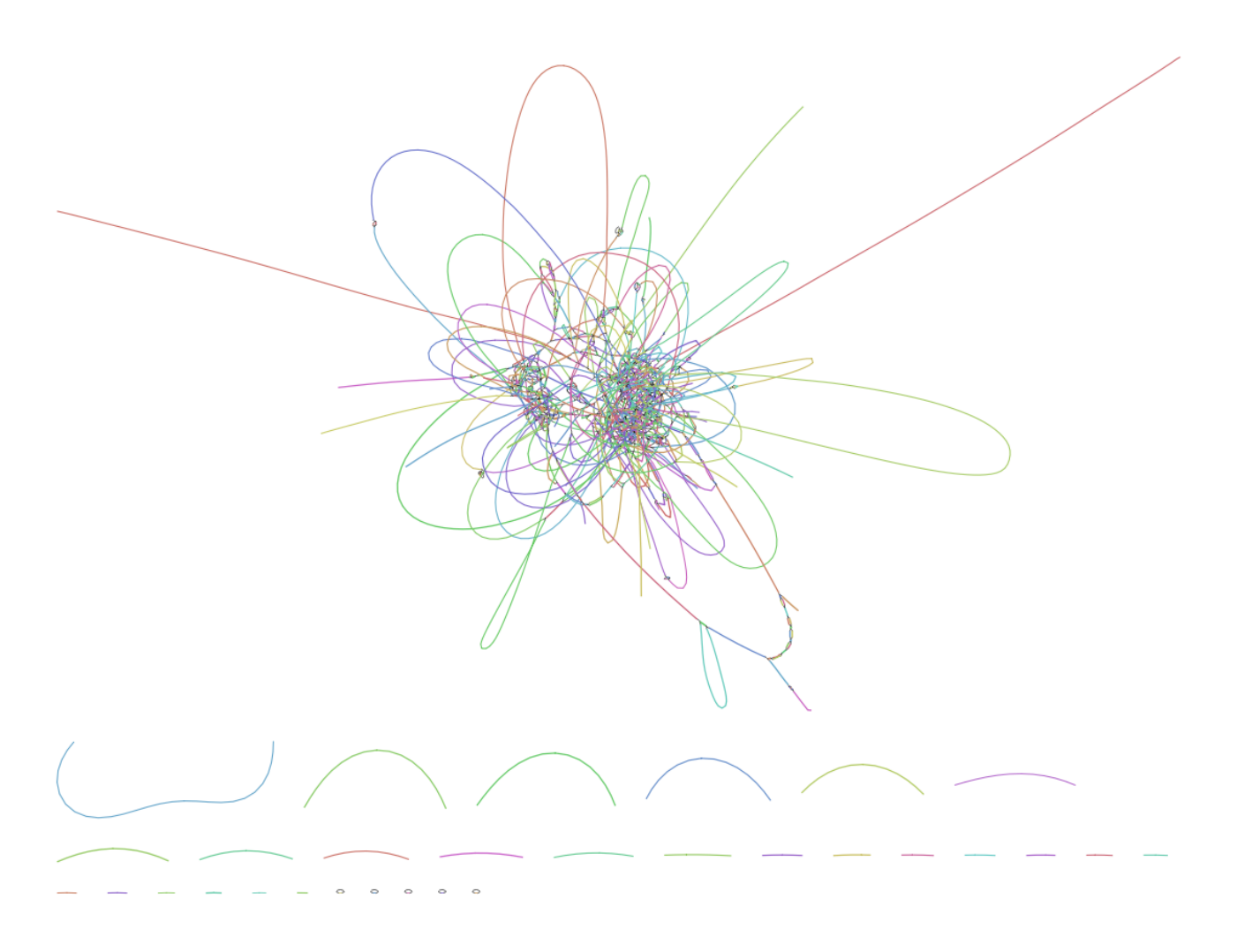
Reads:

```
AGCCTGA
AGCATGA
```

dBG, $k = 3$:

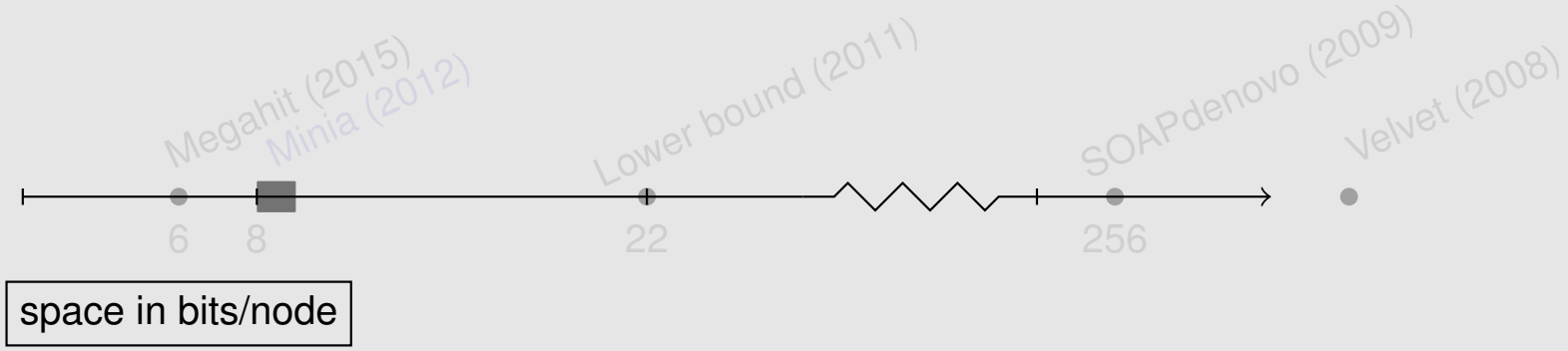


Actual (simplified) de Bruijn graph



chr14:20Mbp-20.5Mbp GAGE PE reads, **SPAdes** 3.8 k=31: 1k nodes

This talk: history of de Bruijn graphs data structures, from a space usage perspective



The early days (2008-2010)

- First genome assemblers for short-read data: EULER-SR, Velvet, SOAPdenovo
- High memory usage: 120 GB for human genome (SOAP)

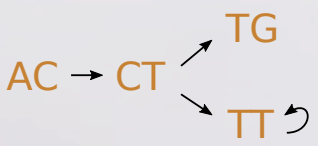
Table 2. Summary of bacterial assemblies using 454 reads

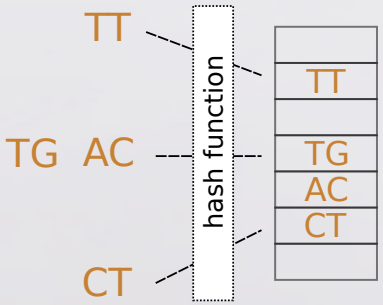
Genome	Assembler	No. long contigs	Total length of long contigs (in kb)	N50 (in bases)
S. pneumoniae	EULER-SR	127	2001	32,619
	Newbler	253	2000	11,905
	Repeat graph	136	2091	36,004
E. coli	EULER-SR	199	4277	46,887
	Newbler	141	4531	60,757
	Repeat graph	94	4560	125,693

Table from Chaisson *et al* 2008

Exact encoding of the de Bruijn graph using a hash table

de Bruijn graph





Hash table

$$\frac{\text{nodes} \times \text{bits/node}}{\text{load factor}} = 28 \text{ bits}$$

$$\frac{4 \times 4}{\sim 0.6} = 28 \text{ bits}$$

The birth of a line of research (2011-2012)

- Conway-Bromage (2011) proposed to encode dBG as bit vector
- Bit vector is of size 4^k, **1**s at positions of k-mers
- Efficient succinct encoding (Okanohara *et al* 2006): $O(nk)$
- Info-theoretically optimal

Exact encoding of the de Bruijn graph using a bit vector

A=00b G=10b
C=01b T=11b

AC=0001b TG=1110b
CT=0111b TA=1100b

0000b	0001b	0010b	0011b	0100b	0101b	0110b	0111b	1000b	1001b	1010b	1011b	1100b	1101b	1110b	1111b
0	1	0	0	0	0	0	1	0	0	0	0	1	0	1	0

16 bits

(Compare with 28 bits using a hash table)

A worst-case lower bound

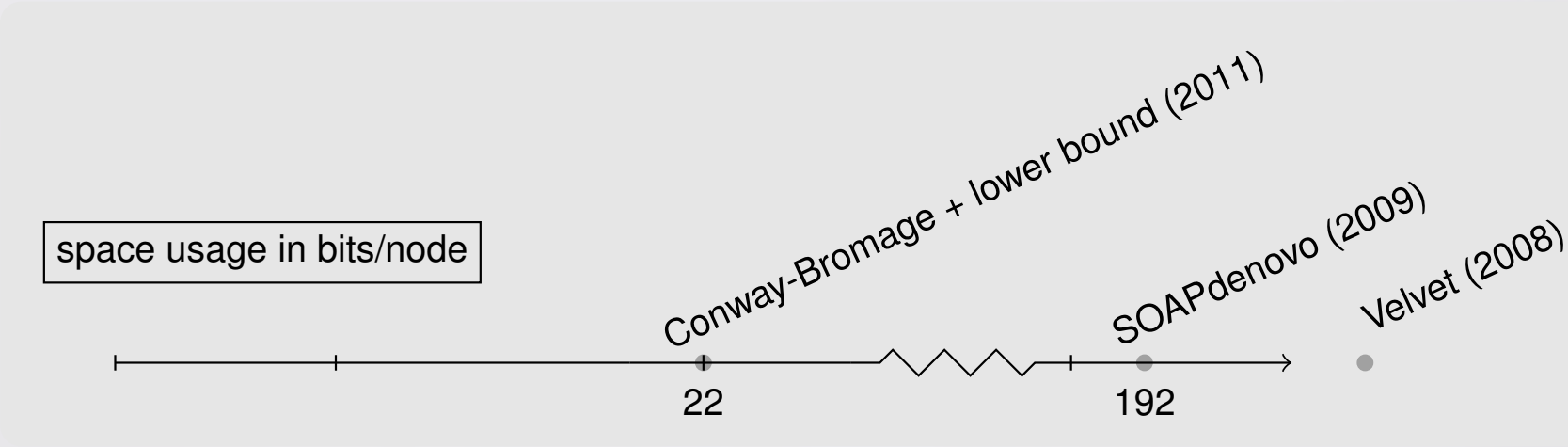
- A de Bruijn graph with n nodes **only needs to records the nodes**
- **Bijection** between **dBG nodesets** and **binary vectors** of length 4^k
- How many bit vectors?

$$\binom{4^k}{n}$$

- Thus, **minimal number of bits** to store a dBG of n nodes:

$$\log_2(\binom{4^k}{n})$$

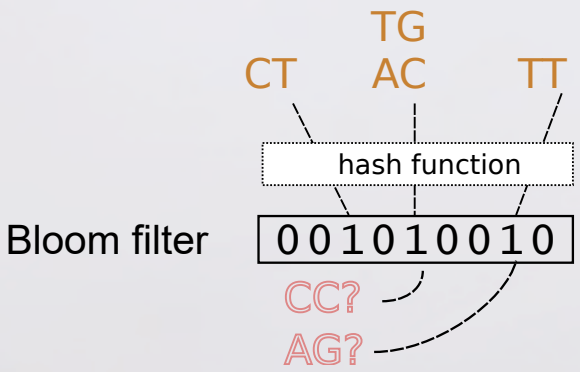
Is this the end of the research line?

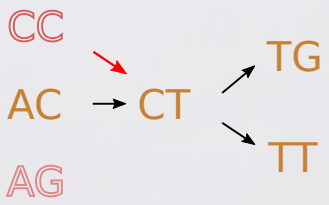


Beating the lower bound (by inexactness, 2012)

- In 2011 Pell *et al* proposed an inexact dBG representation
- bit vector is replaced by a Bloom filter
- In 2012 G. Rizk & I proposed **Minia**
- False positives *k*-mers next to true positives are flagged
- Remaining false positives are ‘inconsequential’

Approximate encoding of the de Bruijn graph using a Bloom filter



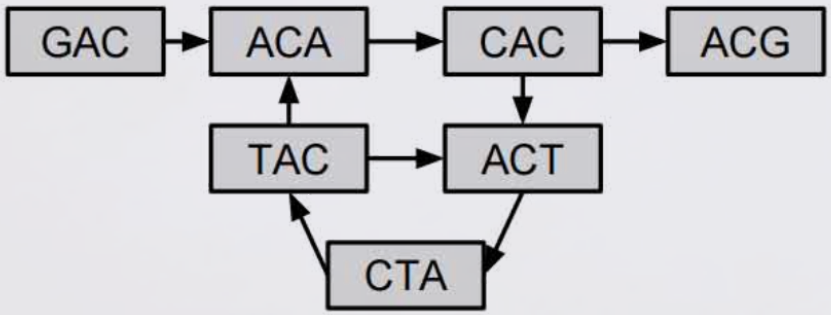


9 bits

(Compare with 16 bits using a bit vector)

Beating the lower bound (by instance specificity, 2012)

- In 2012 Sadakane *et al* proposed the **BOSS** encoding
- Burrows-Wheeler transform modified to store a set of *k*-mers
- Uses **4-6 bits**/*k*-mer but *k*-mers need to have a 'nice' property:
- *k*-mers need to originate from a small number of long strings



An edge-based DBG with *k*=3;

Edges = {GACA, ACAC, CACG, CACT, ACTA, CTAC, TACA, TACT}

	F	Tip	Last		W
A	0	0	1	ACA	C
C	3	1	0	$GA	C
G	8	0	1	CTA	C
T	9	0	0	CAC	G
		0	1	CAC	T
		0	1	GAC	A
		0	0	TAC	A-
		0	1	TAC	T-
		0	1	ACG	$
		0	1	ACT	A

Dummy Edges = { GAC, ACG}

Fig: MEGAHIT

Fun fact: Minia and BOSS both introduced at the same conference (WABI) in 2012

New lower bounds (2014)

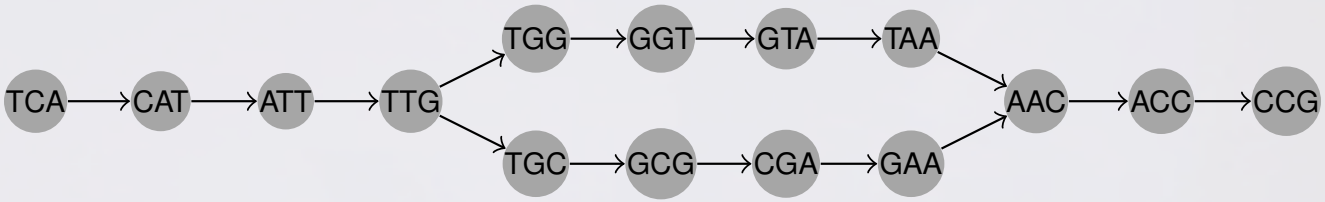
- At RECOMB'14 we showed two new lower bounds[1]:
  - **2 bits**/k-mer as an absolute minimum in the linear case
  - **3.24 bits**/k-mer for navigation in general dBGs
- Side note: no matching upper boud for general dBGs
- Concept: navigational data structures (NDS)

	NDS	Membership (e.g. hash table)
Traverse dBG from known nodes	✓	✓
Query **membership** of arbitrary nodes	*x*	✓
Enumerate nodes	*x*	✓

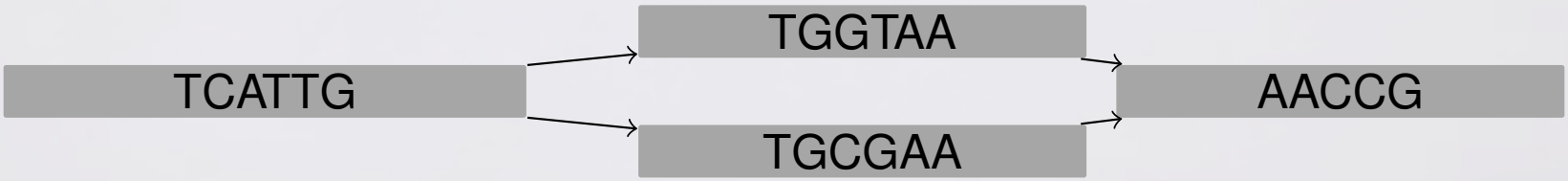
[1] R Chikhi, A Limasset, S Jackman, JT Simpson, P Medvedev, *On the representation of de Bruijn graphs*

Compacted dBG

Input:



After **compaction**, output:



FM-index of the nodes of the compacted dBG: **DBGFM**, 2.34 bits/nodes

Construction algorithms

k-mer counting: **KMC3**, **DSK2**, **Jellyfish2**

Construct a compacted dBG:

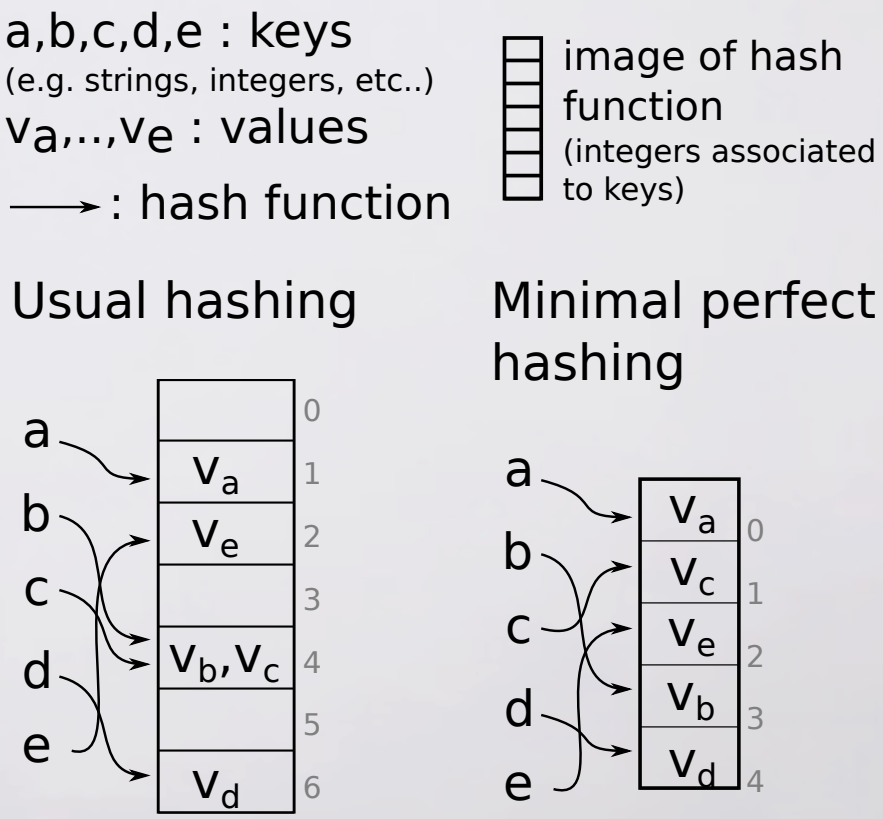
- From reads: **BCALM2**
- From reference genomes: **Cuttlefish**, **TwoPaCo**
- From either: **Bifrost** (also supports queries)

Current state of the art for dBG representation

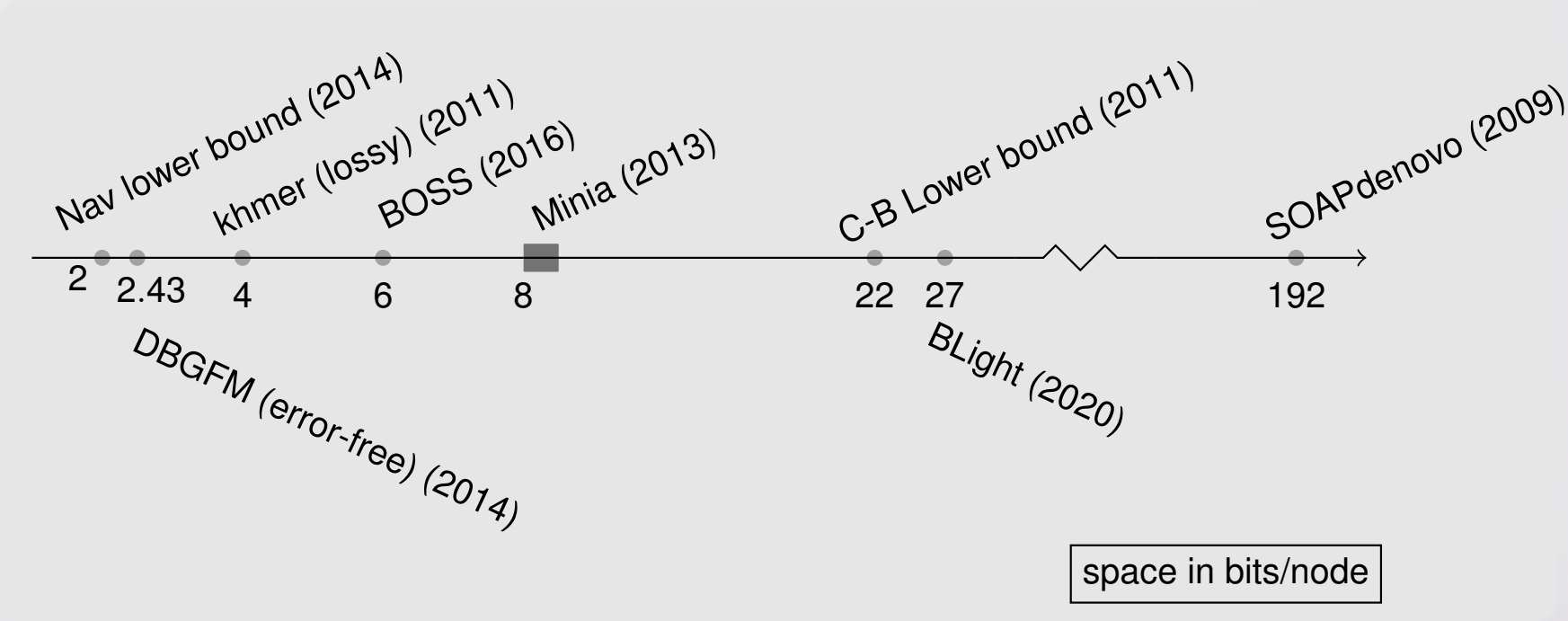
- Minimal perfect hashing (MPHF) library: **BBHash**
- Based on MPHF: **Pufferfish**, **BLight**, **FDBG**
- Based on FM-index/variants: **DBGFM**, **BOSS**, **dynamicBOSS**, **bufBOSS**
- Others: **Bifrost**

Nowadays:

- Focus is less on space, more on features
- Fast query times, associativity, dynamicity



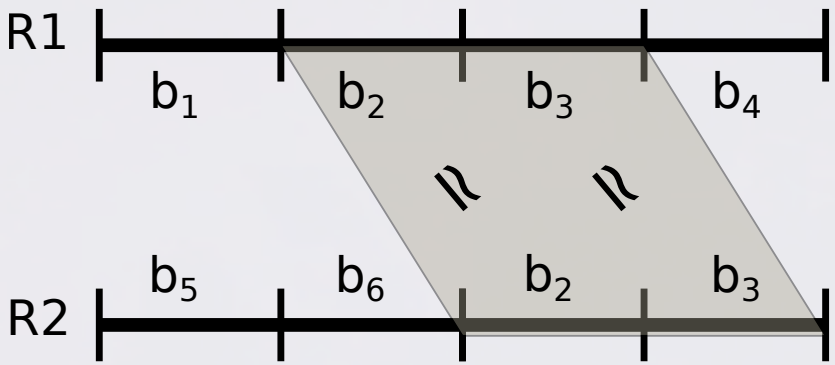
Where are we now?



dBG and long reads

de Bruijn graphs on long reads: `wtdbg2`

1) Reads are binned into 256bp bins



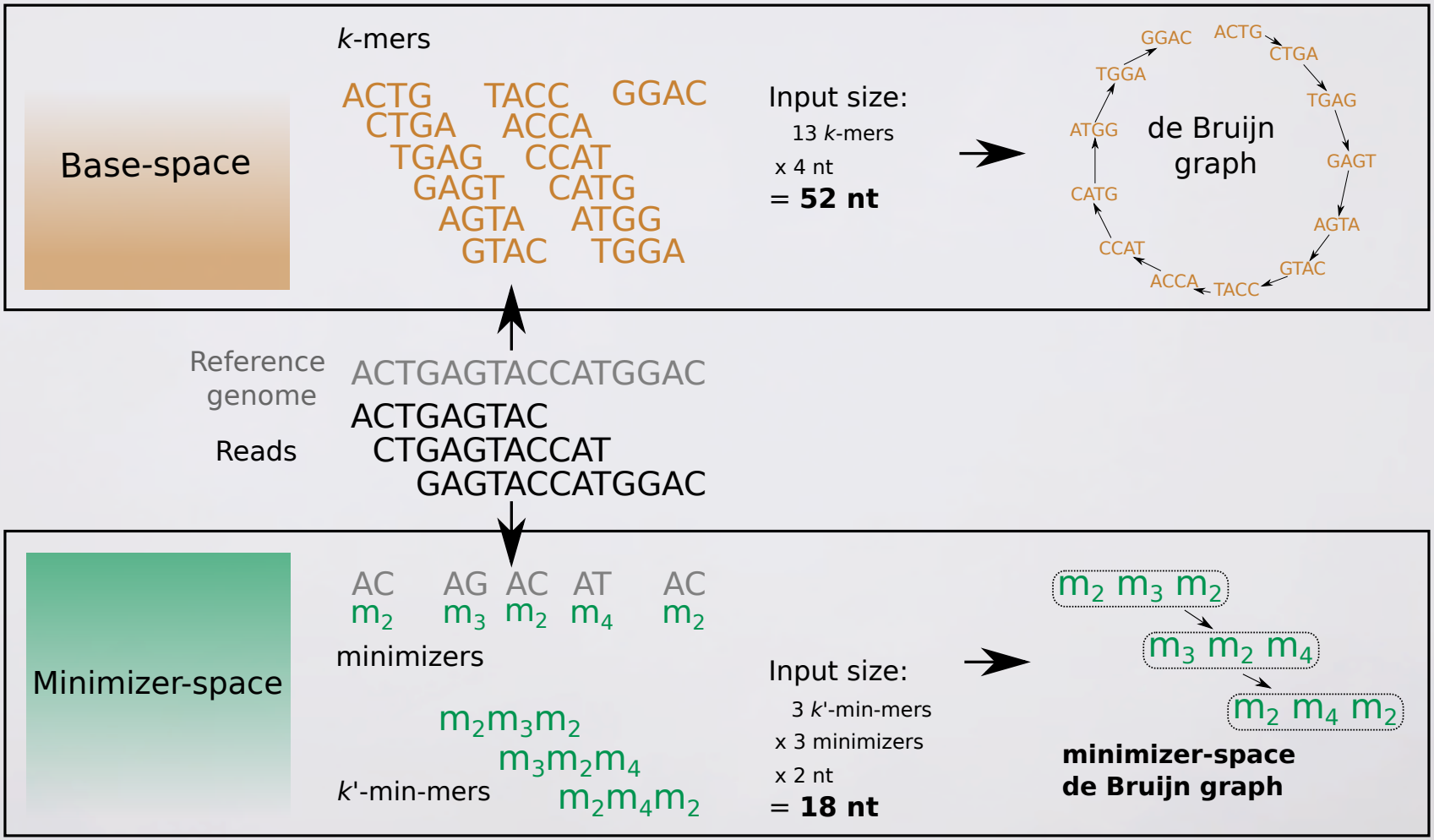
2) Alignments are found between read bins that share k-mers, using Smith-Waterman

3) Bases are forgotten, a "fuzzy" de Bruijn graph is constructed over the bins

$$b_6 b_2 b_3 \rightarrow b_2 b_3 b_4$$

dBGs on long reads: Minimizer-space de Bruijn graphs

(Sneak peek of an upcoming RECOMB'21 talk.)



Conclusion

Caveats

- Only a subset of approaches were presented
- Ignored query times
- Ignored associated info (e.g. k-mer abundances)
- Ignored analysis environment (error-correction, assembly algorithms)
- Ignored multi-k
- Ignored reverse-complements

Future directions

- Representations of **multiple samples**: REINDEER, BIGSI, SBT, HowDeSBT, MetaGraph, etc.. (Marchet *et al* review in Genome Res'20)
- Efficient storage of **abundances**: Italiano *et al*; Shibuya & Kucherov, . . .
- Best adaptation to **long reads**: wtdbg2, mdBG, Flye, . . .
- **Disk** compression: SPSS, Simplitigs, . . .
- A standard file format: `github.com/Kmer-File-Format`

This was an informal take on a more complete review

RESEARCH-ARTICLE OPEN ACCESS

Data Structures to Represent a Set of *k*-long DNA Sequences

Authors: Rayan Chikhi, Jan Holub, Paul Medvedev Authors Info & Affiliations

ACM Computing Surveys, Volume 54, Issue 1 • April 2021 • Article No.: 17, pp 1–22 • https://doi.org/10.1145/3445967

Published: 08 March 2021

Another bioinformatics educational article

Modeling biological problems in computer science: a case study in genome assembly FREE

Paul Medvedev

Briefings in Bioinformatics, bby003, https://doi.org/10.1093/bib/bby003

Published: 30 January 2018 **Article history** ▾

SeqBio Group @ Institut Pasteur

Y. Dufresne, R. Vicedomini, L. Denti, T. Lemane, C. Duitama, L. Blassel

Funding: RiSE Pangaia, ITN Alpaca, ANR Inception, ANR Prairie, ANR Transipedia, ANR SeqDigger

Lex Nederbragt
@lexnederbragt

En réponse à @ctitusbrown

“Finding your way in life is like finding the genome in a De Bruijn graph: it is very easy to find *a* path, very hard to find *the* path”.