Minimizer-space de Bruijn graphs for pangenomics

Rayan Chikhi

Institut Pasteur & CNRS

Compression + Computation, Jan 2022

Maybe you’ve seen this talk before..

Diff with:

- **RECOMB ’21**:
  - More pangenomics
- **Pangenome Bio Hacking, Dec ’21**:
  - More discussion on
    1. minimizers
    2. some philosophical considerations on pangenomics

Application 1: Long read genome assembly

- Oxford Nanopore, PacBio CLR
  - 10-1,000 kbp reads, **5-12**% error rate
- PacBio HiFi
  - 10-25 kbp reads, $\leq$ **1**% error rate

Classical *de Bruijn* graphs not applicable (no long error-free *k*-mers). Instead:

- Overlap graphs (`Canu`, `miniasm`, `Shasta`, `Peregrine`, `hifiasm`, ...)
- Fuzzy dBGs (`wtdbg2`)
- Sparse dBGs: A-Bruijn or minimizers (`Flye`, `MBG`, `LJA`)

Challenge: Approaches don't scale (high resource usage, slow assembly time)!

Application 2: Bacterial Pangenomics: representing and searching in 100,000s bacterial genomes

- *k*-mer indexes (VARI, Bifrost, MetaGraph, Reindeer, SSHash ..)[1]
- MinHash sketches (sourmash)
- ¿Pangenome graphs?

[1] review: Chikhi, Holub, Medvedev 2019, Marchet *et al* 2020

Application 2: Bacterial Pangenomics: representing and searching in 100,000s bacterial genomes

- *k*-mer indexes (VARI, Bifrost, MetaGraph, Reindeer, SSHash ..)[1]
- MinHash sketches (sourmash)
- ¿Pangenome graphs?

Graph challenges:

- terabyte-sized input
- construction
- visualization

[1] review: Chikhi, Holub, Medvedev 2019, Marchet *et al* 2020

Application 2: Bacterial Pangenomics: representing and searching in 100,000s bacterial genomes

- *k*-mer indexes (VARI, Bifrost, MetaGraph, Reindeer, SSHash ..)[1]
- MinHash sketches (sourmash)
- ¿Pangenome graphs?

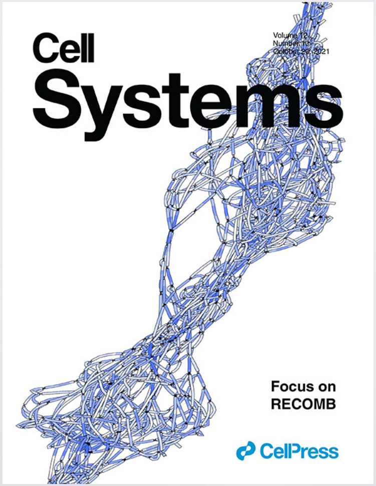
Graph challenges:

- terabyte-sized input
- construction
- visualization

In this talk: 100x-1000x cheaper pangenome graphs through controlled information loss

[1] review: Chikhi, Holub, Medvedev 2019, Marchet *et al* 2020

highly scalable dBGs: Minimizer-space de Bruijn graphs



Barış Ekim

Bonnie Berger

Preliminaries *k*-mers, de Bruijn graph (dBG)

Reference genome

```
ACTGAGTACCATGGAC
```

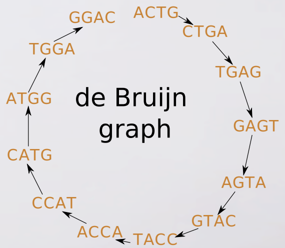
Reads

```
ACTGAGTAC
 CTGAGTACCAT
   GAGTACCATGGAC
```

Base-space

k-mers

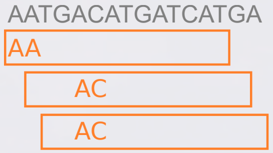
```
ACTG      TACC      GGAC
 CTGA      ACCA
  TGAG      CCAT
   GAGT      CATG
    AGTA      ATGG
     GTAC      TGGA
```



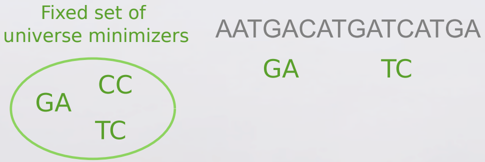
Preliminaries: Minimizers

Two kinds:

- **window.** Local: "smallest" ℓ-mer in a window



- **universe.** Global: set of ℓ-mers with low hash values



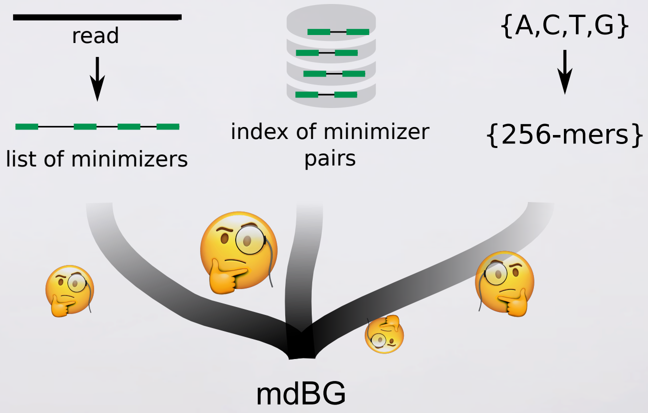
From now on: **universe.** (Also called Scaled MinHash)

This work: stems from three ideas

Shasta
(Shafin et al, 2020)

Peregrine
(Chin et al, 2019)

wtdbg2
(Ruan et al, 2020)



Our approach: Minimizers as *tokens* of the alphabet

Classical alphabet: $\Sigma_{DNA} = \{A, C, T, G\}$

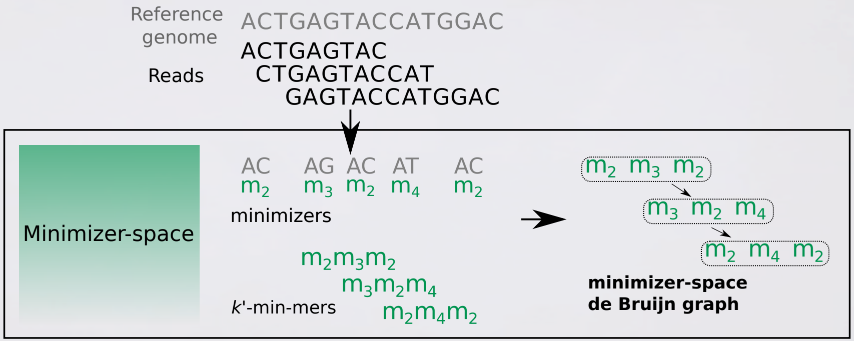
A k-mer with $k = 3$: AGT

Our approach: Minimizers as *tokens* of the alphabet

Classical alphabet: $\Sigma_{DNA} = \{A, C, T, G\}$
A k-mer with $k = 3$: AGT

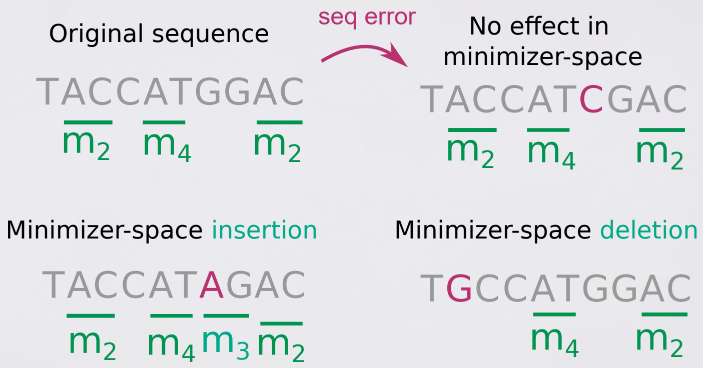
Minimizer alphabet: $\Sigma^{\ell} = \{\text{all minimizers of length } \ell\} = \{m_1, m_2, m_3, \ldots\}$
where e.g. $\ell = 2$, $m_1 = AA$, $m_2 = AC$, $m_3 = AG$, $m_2 = AT$
A k-mer over Σ^{ℓ} (a ***k*-min-mer**): $m_1 m_3 m_2$

Minimizer-space de Bruijn graph



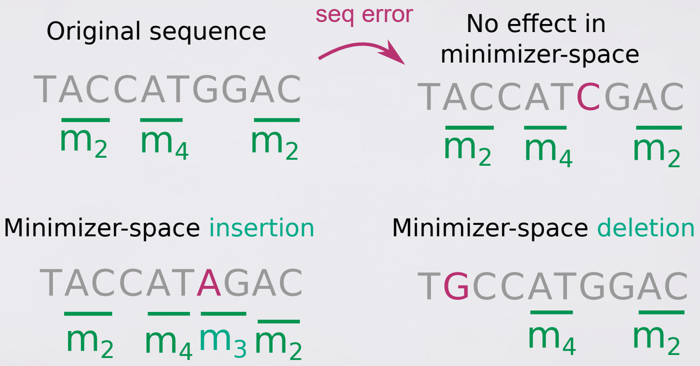
A **minimizer-space de Bruijn graph** is a **de Bruijn graph** over the **minimizer alphabet.**
Nodes = k-min-mers,
Edges = exact overlaps between k-1 minimizers

Sequencing errors propagate to minimizer-space



Error rate: base-space << minimizer-space,
e.g. 5% in base-space corresponds to 50% in minimizer-space.

Sequencing errors propagate to minimizer-space

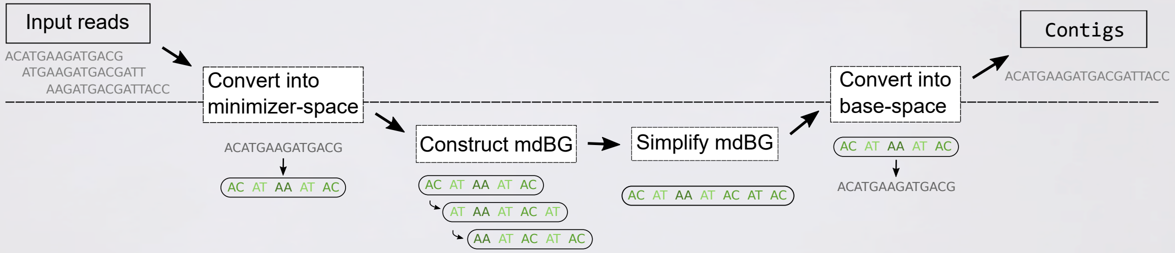


Error rate: base-space << minimizer-space,
e.g. 5% in base-space corresponds to 50% in minimizer-space.

Error correction: minimizer-space POA (base-space POA: Lee *et al*, ‘02))

Applied to whole-genome *de novo* assembly

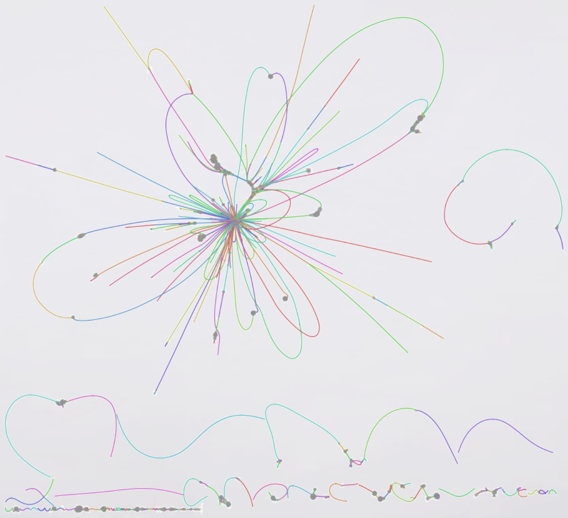
From accurate HiFi (< 1% error-rate) reads



Whole human PacBio HiFi (HG002) 50x coverage:

Tool name	Peregrine	hifiasm	rust-mdbg
Wall-clock time	14h8m	58h41m	**10m23s**
Memory usage	188 GB	195 GB	**10 GB**
# contigs	8109	431	805
NG50 (Mbp)	18.2	88.0	16.1
Genome fraction	97.0%	94.2%	95.5%

Human HiFi mdBG



Assembly implementation details

- `gfatools` (H. Li, unpublished) for graph simplifications
- Automatic parameters (suboptimal):

$$\ell = 12$$

$$\text{density} = 0.003$$

$$k = \frac{3}{4} \cdot \text{avg\_readlen} \cdot \text{density}$$

- Multi-k script (à la IDBA/SPAdes).
- Code available at `github.com/ekimb/rust-mdbg/`

Minimizer considerations

- We used universe minimizers, computed using NtHash (rolling).
- Aware of: syncmers, strobemers (untested), LCP
- Barış' insight: we need MCAS's (substrings that align confidently)

Density minimizers vs syncmers in mDBG

D. mel 100x	**Density minimizers**	**Downsampled syncmers**
Best N50	3.9 Mbp	3.8 Mbp
Asm size	111 Mbp	111 Mbp
k	30	25
l	10	10
s	N/A	6
density	0.0035	0.02

Notes:
- Best result out of a coarse parameter grid search
- Both schemes use same hash function
- LCP: in journal (similar)

Results: Metagenome assembly

Zymo D6331 mock metagenome HiFi

Species	**Abundance**	`hifiasm`	`rust-mdbg`
A. muciniphila	1.36%	100.000%	100.000%
B. fragilis	13.13%	99.994%	99.997%
B. adolescentis	1.34%	100.000%	99.730%
C. albican	1.61%	67.832%	39.821%
C. difficile	1.83%	99.996%	99.978%
C. perfringens	0.00%	0.005%	0.005%
E. faecalis	0.00%	0.006%	0.006%
E. coli B1109	8.44%	100.000%	97.918%
E. coli b2207	8.32%	100.000%	98.663%
E. coli B3008	8.25%	100.000%	99.558%
E. coli B766	7.83%	96.913%	96.270%

Species	**Abundance**	`hifiasm`	`rust-mdbg`
E. coli JM109	8.37%	100.000%	97.852%
F. prausnitzii	14.39%	100.000%	100.000%
F. nucleatum	3.78%	100.000%	99.960%
L. fermentum	0.86%	100.000%	100.000%
M. smithii	0.04%	99.840%	87.175%
P. corporis	5.37%	99.561%	99.561%
R. hominis	3.88%	100.000%	100.000%
S. cerevisiae	0.18%	69.522%	39.556%
S. enterica	0.02%	6.232%	4.619%
V. rogosae	11.02%	100.00%	100.000%

	`hifiasm`	`rust-mdbg`
Running time	34h29m	**55s**
Memory usage	83 GB	**0.9 GB**

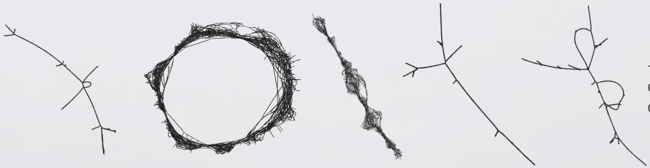
Results: Pangenome graph of 661,405 bacterial genomes

Data from Blackwell et al, 2021:

```
2.9T 661k_assemblies.fa
1.6T 661k_assemblies.fa.lz4
```

```
rust-mdbg -k 10 -l 12 --density 0.001 --minabund 1 661k_assemblies.fa.lz4
```

Largest 5 connected components:



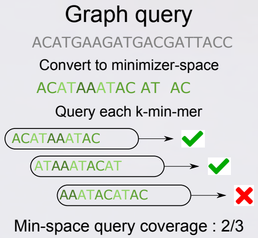
+ 725,820 connected components

Taxons in component	18	22	4	22	10
Dominant species	*Mycobacterium tuberculosis*	*Salmonella enterica*	*Burkholderia gladioli*	*Pseudomonas protegens*	*Cupriavidus alkaliphilus*

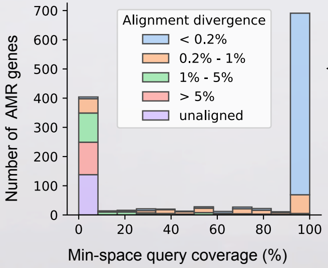
Biological results: Querying AMR genes

AMR genes database

1,279 genes (AMRFinderPlus)



Retrieval of AMR genes



Behind the scenes of mdBG pangenome construction

- `rust-mdbg` tool: from reads to raw mdBG
- set of scripts (`github.com/ekimb/rust-mdbg/tree/master/experiments/661k_genomes`)

In particular:

- `grep` for k-min-mers search (10mins)
- `lz4` k-min-mer compression
- pangenome `.gfa.gz` just the topology: 2-20GB
- "Resolution": 10-100kbp (kminmer span)

Behind the scenes of mdBG pangenome construction

- `rust-mdbg` tool: from reads to raw mdBG
- set of scripts (`github.com/ekimb/rust-mdbg/tree/master/experiments/661k_genomes`)

In particular:

- `grep` for k-min-mers search (10mins)
- `lz4` k-min-mer compression
- pangenome `.gfa.gz` just the topology: 2-20GB
- "Resolution": 10-100kbp (kminmer span)

What we *don't* have:

- Succinct (colored) mdBGs
- O(1) k-min-mer sequence search
- visualization of pangenome mdBGs (100k-1M nodes)

FAQ on mdBGs

Short reads?

- Doesn’t seem applicable

Improving assembly N50?

- Better graph simplifications

Nanopore data?

- 5% error rate is too much, but 1% sounds promising

Towards bigger and bigger pangenomes..

Community explores many directions:

1. Sketches
   - `Mash, sourmash`
   - Low-resolution search, ~~graph~~
2. All nucleotides
   - Approx: `BIGSI, HowDeSBT`
   - Exact: `Cuttlefish 2, MetaGraph, vg, minigraph, ..`
   - High-resolution search, graph
   - Expensive to store
3. “In-between”
   - `mdBG`
   - Low-resolution search, graph
   - Inexpensive to store

Towards bigger and bigger pangenomes..

Community explores many directions:

1. Sketches
   - `Mash, sourmash`
   - Low-resolution search, ~~graph~~
2. All nucleotides
   - Approx: `BIGSI, HowDeSBT`
   - Exact: `Cuttlefish 2, MetaGraph, vg, minigraph, ..`
   - High-resolution search, graph
   - Expensive to store
3. “In-between”
   - `mdBG`
   - Low-resolution search, graph
   - Inexpensive to store
4. Gene families
   - What biologists actually do
   - Lowest-resolution, ~~seq search~~, sometimes graph
   - Inexpensive to store

Some open questions

1. Can one represent all life 31-mers?
2. Can one represent all life 31-mers up to 2 edit mutations?
3. Can one represent all life k-min-mers? (k=10, l=12, density to be determined)

4. Can one represent all prokaryote+viral 31-mers known to date?
5. Can one represent all human 31-mers known to date?

Conclusion

- **mdBGs** can not only perform genome assembly but also represent pangenome graphs for large collections (661k bacterial genomes) efficiently (10s of GB) at 10-100kbp resolution

Main idea: “*k*-mers” over sequences of minimizers “characters”

Potential hacking directions (cont’d):

- Higher-resolution mdBGs (1 kbp span for *k*-min-mers)
- Pangenome mdBGs for eukaryotes
- Automated differential analysis on colored GFAs
- Large structural variant calling on colored GFAs