Some ingredients for de novo (meta)genomic assembly

Rayan Chikhi

CNRS
Univ. Lille 1, France

BiATA'17

Context

Motivation: **sequence graph representation** of

1. populations, pan-genomes
2. (pooled) (meta)genome assembly
3. transcriptomes
4. variant detection
5. 3rd generation reads

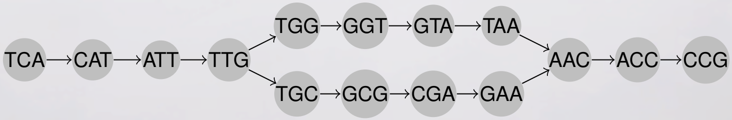
"Old" concepts, improved techniques:

1. de Bruijn graph construction & representation
2. simplifications, multi-k

de Bruijn Graph

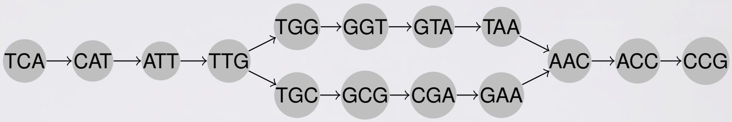
```
sequences:  TCATTGGTAACCG
            TCATTGCGAACCG

  k-mers:   TCA
   (k=3)     CAT
              ATT
               ...
```

nodes: *k*-mers (words of length *k*)
edges: exact suffix-prefix overlaps of length $k - 1$

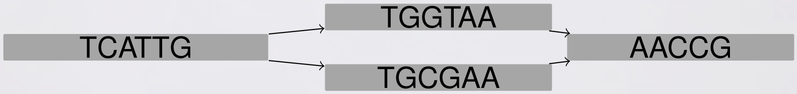


Compacted de Bruijn Graph

(non-compacted) de Bruijn graph:



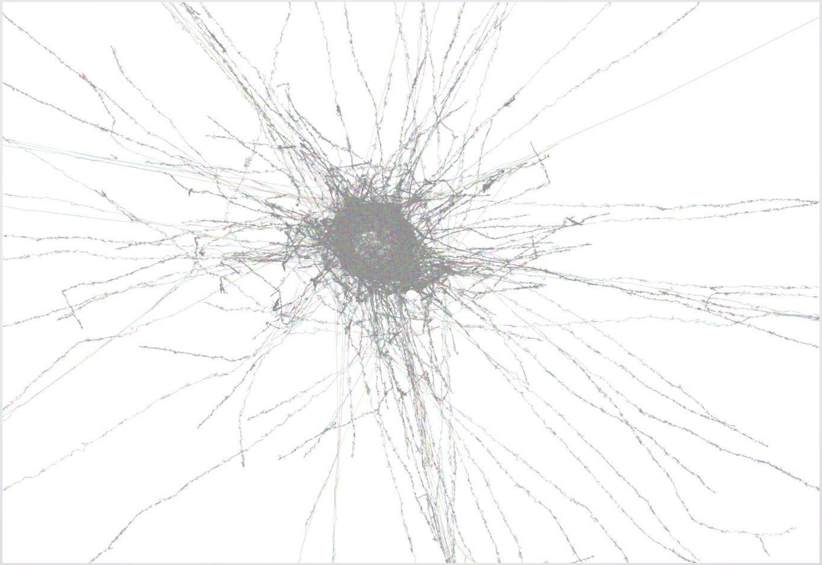
Compacted de Bruijn graph:



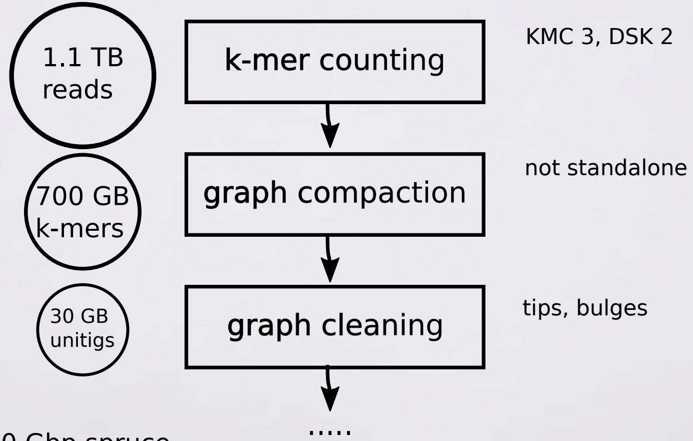
Maximal non-branching paths become single nodes (*unitig*).

- no loss of information, less space
- can "simulate" a de Bruijn graph
- not dynamic
- unitigs are more specific than *k*-mers

Real data


dBG of S. aureus bacterium (SRR022865), uncleaned compacted graph

de Bruijn graph-based tools



20 Gbp spruce
[Birol 2013]

Ingredient 0

$$\begin{tikzcd} \text{fastq} \arrow[r] & \text{DSK 2} \arrow[r] & \text{kmers} \end{tikzcd}$$

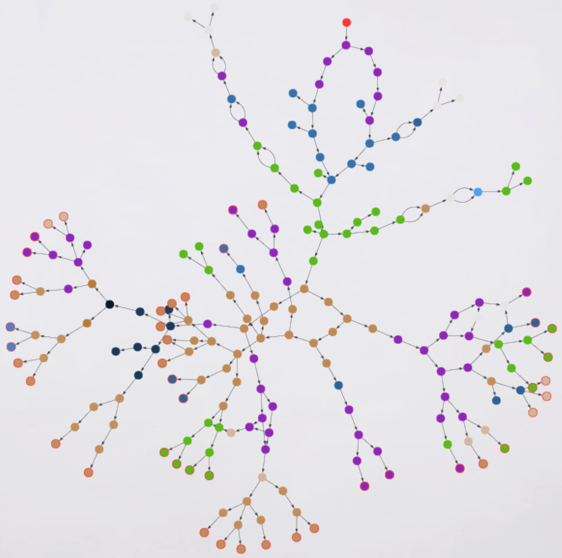
k-mer counter, minimum substring partitioning (MSPKmerCounter, KMC 3). [Rizk *et al*, 2013]

minimizer of *s*:
smallest ℓ-mer in *s*
[Roberts *et al*, 2004]

```
TGACGGG
 GACGGGT
  ACGGGTC
   CGGGTCA
    GGGTCAG
     GGTCAGA
```

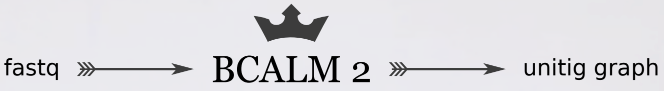
- *k*-mers *partitioned by minimizer*
- integrated in GATB library
  D. Lavenier's talk
- 30 mins, 1 GB mem per genome Gbp

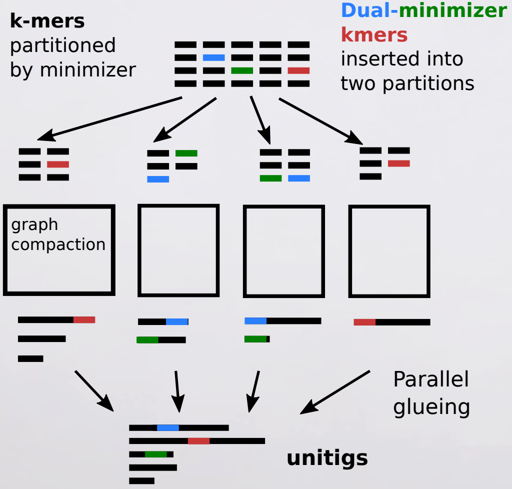
Proximity of minimizers on the dBG



using pyGATB

Ingredient 1




k-mers partitioned by minimizer
Dual-minimizer kmers inserted into two partitions
graph compaction
Parallel glueing
unitigs

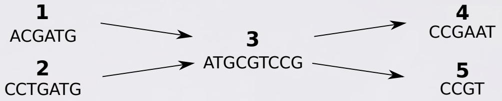
[Chikhi, Limasset, Medvedev ISMB'16]

1 hour, 2 GB memory per genome Gbp

outputs GFA

Ingredient 2: data structure for unitigs

Motivation: Bloom Filter, XBWT, FM-index compress well but slow navigation.



list of neighbors

out: **3 3 4 5** in: **3 3** ~16 bytes/unitig

γ-encoded numbers of neighbors

out: **1 1 2 0 0** in: **0 0 2 1 1** ~1.25 bytes/unitig

concatenated unitigs

ACGATG[;]CCTGATG[;]ATGCGTCCG[;]CCGAAT[;]CCGT 2bits/nt

γ-encoded unitigs lengths

6 7 9 6 4 0.25 log(n) bytes/unitig

mean abundance, flags, etc..

1.25 bytes/unitig

Size: 300 extra bits/unitig

human dBG: **3.1** GB

`dbgfm`: 2*k* extra bits/unitig

Minia (bits/kmer): 8

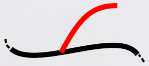
succinct DBG: 2 – 16

deBGR: 25 (incl. counts)

Related: `xg` succinct graphs from the `vg` toolbox, but immutable

Ingredient 3

$$\text{unitigs} \longrightarrow \textbf{Minia 3} \longrightarrow \text{contigs}$$



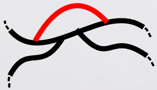
Tip removal:

$len_{tip} \leq 3.5k$

or

$len_{tip} \leq 10k$

$2cov_{tip} \leq cov_{neighbors}$



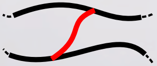
Bulge removal:

$len_{bulge} \leq \max(3k, 100)$

$cov_{bulge} \leq 1.1cov_{altpath}$

$len_{altpath} = len_{bulge} \pm delta$

$delta = \max(0.1len_{bulge}, 3)$



Erroneous connection removal:

$len_{EC} \leq 10k$

$4cov_{EC} \leq cov_{neighbors}$

SPAdes-inspired graph simplifications

entirely command-line parameterizable

GFA in, GFA out

Dealing with a flood of erroneous *k*-mers

. . . and keeping low-coverage, good *k*-mers.

The MEGAHIT way: abundance cut-off at 2, *mercy* *k*-mers

The SPAdes way: abundance cut-off at 1, then $(k+1)$-mers pre-simplifications

New components:

1. pre-simplifications inside BCALM 2
2. stand-alone fixed-memory tip clipping software (BTRIM) [Limasset, unpublished]
3. stand-alone mercy *k*-mers module [unpublished]

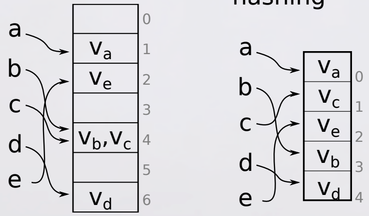
Ingredient 4: BBHash

a,b,c,d,e : keys
(e.g. strings, integers, etc..)
$v_a,..,v_e$: values
$\longrightarrow$: hash function

image of hash function
(integers associated to keys)

Usual hashing

Minimal perfect hashing



Static *k*-mer index, but not only

C++: `#include "BooPHF.h"`

3 bits/key

construction: 10^9 keys/min, ˜0 space overhead

tested on 10^{12} keys

SPAdes ♡

`github.com/rizkg/BBHash`, [Rizk et al, SEA 2017]

Performance

BCALM 2

20 Gbp spruce genome

1 TB reads
9 hours
31 GB memory

Minia 3

32 Gbp axolotl genome

10x coverage
1.3 kbp N50
1 week assembly time
[unpublished]

Modularity

Precursors: ABySS for dBG; SGA and miniasm for string graphs

fastq → DSK 2 → kmers

fast, minimizer-partitioned output, integrated in GATB

`github.com/GATB/dsk`

fastq → BCALM 2 → unitig graph

outputs GFA, mean abundance per unitig

`github.com/GATB/bcalm`

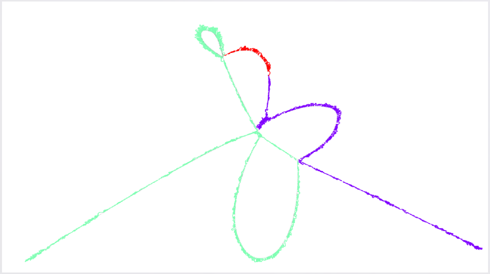
unitigs → Minia 3 → contigs

GFA in/out, parameterized graph simplifications

`github.com/GATB/minia`

Poster

Debugging long-read genome assemblies using string graph analysis



Canu assembled contigs projected onto minimap's overlap graph

w/ JS Varré, P. Marijon