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:  TCATTGGTAAACCG
            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:

\[
\begin{align*}
TCA & \rightarrow \text{CAT} & \rightarrow \text{ATT} & \rightarrow \text{TTG} \\
TGG & \rightarrow \text{GGT} & \rightarrow \text{GTA} & \rightarrow \text{TAA} \\
TGC & \rightarrow \text{GCG} & \rightarrow \text{CGA} & \rightarrow \text{GAA} \\
AAC & \rightarrow \text{ACC} & \rightarrow \text{CCG}
\end{align*}
\]

Compacted de Bruijn graph:

\[
\begin{align*}
\text{TCATTG} & \rightarrow \text{TGGTAA} \\
\text{TGC} & \rightarrow \text{CGA} & \rightarrow \text{GAA} & \rightarrow \text{GAA} \\
\text{AAC} & \rightarrow \text{ACC} & \rightarrow \text{CCG}
\end{align*}
\]

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

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

dBG of S. aureus bacterium (SRR022865), uncleared compacted graph
de Bruijn graph-based tools

1.1 TB reads

700 GB k-mers

30 GB unitigs

k-mer counting

graph compaction

graph cleaning

......

KMC 3, DSK 2

not standalone

tips, bulges

20 Gbp spruce [Birol 2013]
fastq \xrightarrow{} \text{DSK 2} \xrightarrow{} \text{kmers}

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

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

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

TGACGGG
GACGGGT
ACGGGTC
CGGGTCA
GGGTCAG
GGTCAGA
Proximity of minimizers on the dBG
**Ingredient 1**

**fastq** → **BCALM 2** → **unitig graph**

**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.

![Diagram of unitigs]

- **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**: 2k 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
Tip removal:
\[ \text{lentip} \leq 3.5k \]
or
\[ \text{lentip} \leq 10k \]
\[ 2\text{covtip} \leq \text{covneighbors} \]

Bulge removal:
\[ \text{lenbulge} \leq \max(3k, 100) \]
\[ \text{covbulge} \leq 1.1\text{covaltpath} \]
\[ \text{lenaltpath} = \text{lenbulge} \pm \delta \]
\[ \delta = \max(0.1\text{lenbulge}, 3) \]

Erroneous connection removal:
\[ \text{lenEC} \leq 10k \]
\[ 4\text{covEC} \leq \text{covneighbors} \]

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..)
va, .., ve : values

→ : hash function

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 \(\rightarrow\) **DSK 2** \(\rightarrow\) kmers

fast, minimizer-partitioned output, integrated in GATB

[github.com/GATB/dsk](https://github.com/GATB/dsk)

fastq \(\rightarrow\) **BCALM 2** \(\rightarrow\) unitig graph

outputs GFA, mean abundance per unitig

[github.com/GATB/bcalm](https://github.com/GATB/bcalm)

unitigs \(\rightarrow\) **Minia 3** \(\rightarrow\) contigs

GFA in/out, parameterized graph simplifications

[github.com/GATB/minia](https://github.com/GATB/minia)
Debugging long-read genome assemblies using string graph analysis

Canu assembled contigs projected onto minimap’s overlap graph

w/ JS Varré, P. Marijon