Question: is *de novo* genome assembly a solved problem with long reads, yet?

Rayan Chikhi

Institut Pasteur & CNRS

CGSI 2019
Answer: No
Answer: No

Thank you! Any questions?

Acknowledgements: Pierre Marijon, Jean-Stéphane Varré, Antoine Limasset, Camille Marchet, Sergey Nurk, Marco Previtali, Paul Medvedev, Shaun Jackman, Guillaume Rizk, Adam Phillippy
Hello

- New group leader at Institut Pasteur, Paris
- *Algorithms and data structures for biological sequences*

Contributions:
- Methods around $k$-mers
  - Minia, KmerGenie, DSK, BCALM2
- Genome assemblies

@RayanChikhi on Twitter
http://rayan.chikhi.name
Genome assembly

**genome** (unknown)

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

**assembly**

hypothesis of the genome

read

contig
The many applications of assembly

- **Reconstruct** genome
  - transcriptome
  - metagenome
  - genes from different taxa
- Find novel *insertions*
- **SNPs** in non-model organisms
- **cell-free DNA** SVs
- Pangenomics
- ...
(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.”

(Adapted from A. Phillippy’s talk, RECOMB-Seq’19)
A short algorithmic history of genome assembly
- Shortest Common Superstring
- Greedy algorithms
A short algorithmic history of genome assembly

- Shortest Common Superstring
- Greedy algorithms
- String graphs and de Bruijn graphs, both introduced at DIMACS in 1994

A History of DNA Sequence Assembly, G. Myers, 2016
Modern genome assembly: graphs

1. Construct a graph
2. Nodes are reads (or $k$-mers)
3. Edges are overlaps

Theorists will say.. [Nagarajan 09]

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

\[
\begin{align*}
\text{GAT} & \rightarrow \text{ATT} & \rightarrow & \text{TTA} & \rightarrow & \text{TAC} & \rightarrow & \text{ACA} & \rightarrow & \text{CAA} \\
\rightarrow & \text{CAT} & \rightarrow & & & & & & & \\
\end{align*}
\]
Genome assembly is **NP-hard**. [Medvedev, Brudno 2007]

If all **repeats** are **longer** than reads, Genome assembly is **polynomial**. (!) [Nagarajan, Pop 2009]

If all repeats are either **shorter** than reads, or are **spanned** by reads, Genome assembly is **polynomial**, and with a **unique** solution. [Nagarajan, Pop 2009] [Bresler, Bresler, Tse 2013]
But, in practice

- **Illumina** data: none of the previous formalisms applied
- Because graph often disconnected
- Contigs = unambiguous paths in graph

- **Long reads**: bridge between theory and practice appears possible
- HINGLE assembler  

  [Kamath et al, 2017]
Recommended reading

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

Paul Medvedev

*Briefings in Bioinformatics*, bby003, [https://doi.org/10.1093/bib/bby003](https://doi.org/10.1093/bib/bby003)

**Published:** 30 January 2018  **Article history**
Vertebrate/human genome assembly

Outlook:
- Vertebrates: some have high repeat %
- Diploidicity: still badly handled
- Humans: Telomere2telomere within 2 years
Today: supposedly easy cases

- Small (e.g. bacterial) genomes
- Haploid
- Can we *at least* assemble *that* well now?
Genome assembly software is complex

- Coding:
  - PhD
  - or a team of engineers (1-2 years)
- Several not-always-independant components

- Heuristics everywhere

A good genome assembler is like a good sausage, you would rather not know what is inside

(apocryphal) S. Gnerre, ALLPATHS assembler
Long-read genome assemblers

- HGAP
- Canu
- Falcon
- miniasm
- Unicycler
- MARVEL
- Tulip
- ABruijn
- Hinge
- Flye
- Ra
- Wutabaga2
- Shasta
- Peregrine
- ...

...
One chromosome = one contig?

Assembly graph of the *E. coli* genome [Koren, Phillippy 2015]:

Slides adapted from P. Marijon, RECOMB-Seq’19
## NCTC 3000 database

<table>
<thead>
<tr>
<th>Species</th>
<th>Strain</th>
<th>Sample</th>
<th>Runs</th>
<th>Automated Assembly</th>
<th>Manual Assembly</th>
<th>Manual Assembly Chromosome Contig Number</th>
<th>Manual Assembly Plasmid Contig Number</th>
<th>Manual Assembly Unidentified Contig Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achromobacter xylosoxidans</td>
<td>NCTC10807</td>
<td>ERS451415</td>
<td>ERR550491 ERR550506</td>
<td>Pending</td>
<td>EMBL</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Budvicia aquatica</td>
<td>NCTC12282</td>
<td>ERS462988</td>
<td>ERR581162</td>
<td>Pending</td>
<td>EMBL</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Campylobacter jejuni</td>
<td>NCTC11351</td>
<td>ERS445056</td>
<td>ERR550473 ERR550476</td>
<td>Pending</td>
<td>EMBL</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cedecea neteri</td>
<td>NCTC12120</td>
<td>ERS462978</td>
<td>ERR581152 ERR581168</td>
<td>Pending</td>
<td>EMBL</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Citrobacter amalonaticus</td>
<td>NCTC10805</td>
<td>ERS485850</td>
<td>ERR601566 ERR601575</td>
<td>Pending</td>
<td>EMBL</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Citrobacter freundii</td>
<td>NCTC9750</td>
<td>ERS485849</td>
<td>ERR601559 ERR601565</td>
<td>Pending</td>
<td>EMBL</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Citrobacter koseri</td>
<td>NCTC10849</td>
<td>ERS473430</td>
<td>ERR581173</td>
<td>Pending</td>
<td>EMBL</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Corynebacterium diphtheriae</td>
<td>NCTC11397</td>
<td>ERS451417</td>
<td>ERR550510</td>
<td>Pending</td>
<td>EMBL</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cronobacter sakazakii</td>
<td>NCTC11467</td>
<td>ERS462977</td>
<td>ERR581151 ERR581167</td>
<td>Pending</td>
<td>EMBL</td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

599 / 1136 (34 %) assemblies are not single-contig (Feb 2019)
Example (simulated)

- **Dataset**: *T. roseus* (bacteria), simulated PacBio 20x
- **Assembly**: Canu 1.7

Resulting assembly graph:

Can we recover missing edges between contigs?
Not even a repetition problem..

*Dotplot of T. roseus genome against itself.*

Genome has a 460 kbp tandem repeat. It explains only 1 of the 2 contigs breaks.
Let’s have a look at the original overlap graph:
- nodes → reads
- edges → overlaps

Overlap graph (constructed by Minimap2), reads colored by Canu contig.
KNOT: Pipeline

Input

- Assembly contigs
- Raw reads

Contig classification

Raw string graph

Inter-contigs paths search

Augmented assembly graph

Analysis explain before

Output

P. Marijon et al, Bioinformatics 2019
The Augmented Assembly Graph

undirected, weighted graph:
- nodes: contigs extremities
- edges:
  - between extremities of a contig (weight = 0)
  - paths found between contigs (weight = path length in bp)

From

```
From
--------
|        |
|        |
|        |
```

tig 1

tig 4

tig 8

To

```
To
--------
|        |
|        |
|        |
```

tig1

491922

tig8

755235

ovl

tig4
KNOT finds hidden connections between contigs

Across 38 datasets:

<table>
<thead>
<tr>
<th>Mean number of</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Canu contigs</td>
<td>4.32</td>
</tr>
<tr>
<td>Dead-ends in Canu contig graph</td>
<td>4.94</td>
</tr>
<tr>
<td>Dead-ends in AAG</td>
<td>2.70</td>
</tr>
</tbody>
</table>
- AAG’s are generally complete
- Hamilton walks can be enumerated
- Walk weight: sum of edges weights
- **lowest-weight walk** assumed to be the true genome

- Green walk weight: 18,769 bases
- Blue walk weight: 136,229 bases
- Bacterial genome assembly isn’t fully solved
- **Augmented Assembly Graphs** can help

https://gitlab.inria.fr/pmarigon/knot

@pierre_marijon

- Other analysis tool not based on graphs:
- https://github.com/johnomics/tapestry
How robust are genome assemblers?

Simulates reads with
- chimeras
- low-quality regions
- systematic basecalling errors
- ...

https://github.com/rrwick/Badread
Unit test for a single function

```python
def AddTest:
    assert(add(1,1) == 2)
```

Functional test for a whole feature

```python
def MapTest:
    r = mapRead("ACTGATG", genome)
    assert(r.position = 100000)
    assert(r.mapping_length = 150)
    ...
```
Assembly "functional testing"

github.com/rrwick/Long-read-assembler-comparison/

- Bacterial genome (C. kerstersii)
  - 4 tandem copies of rRNA operon, 20kbp repeat
- Simulated PacBio reads, various
  - Read lengths
  - Error rates
  - Coverages
  - …
Reproducing that benchmark

- Same dataset as the previous benchmark
- Simulated reads using another program (PaSS, 2019)
- 50x depth
- Should assemble fine into 1 contig (..maybe?)

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Same assemblers
Reproducing that benchmark

- Same dataset as the previous benchmark
- Simulated reads using another program (PaSS, 2019)
- 50x depth
- Should assemble fine into 1 contig (..maybe?)

- Same assemblers

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td>1 😊</td>
<td>1 😊</td>
<td>2 😞</td>
<td>3 😞</td>
<td>2 😞</td>
</tr>
</tbody>
</table>
Is it a coverage problem?
Is it a coverage problem?

- from 50x coverage

- to 100x coverage

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is it a coverage problem?

- from 50x coverage

- to 100x coverage

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td>1 😊</td>
<td>1 😊</td>
<td>2 😕</td>
<td>3 😕</td>
<td>3 😕</td>
</tr>
</tbody>
</table>

- So, no
Robustness to coverage drops

- Taking the 50x coverage dataset again
- Simulating a coverage drop to 10x somewhere

Before:

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Robustness to coverage drops

- Taking the 50x coverage dataset again
- Simulating a coverage drop to 10x somewhere

Before:

After:

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td>😊</td>
<td>😊</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
</tbody>
</table>

- Assemblers: all unphased by that drop
Robustness to heavy coverage drop

- Dropping down to 5x 😈

10x:

5x:

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Good job Flye!
Robustness to heavy coverage drop

- Dropping down to **5x 😈**

10x:

5x:

<table>
<thead>
<tr>
<th></th>
<th>Canu</th>
<th>Flye</th>
<th>Ra</th>
<th>Unicycler</th>
<th>Wtdbg2</th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td>2 😞</td>
<td>1 😊</td>
<td>3 😞</td>
<td>3 😞</td>
<td>2 😞</td>
</tr>
</tbody>
</table>

- Good job Flye!
Conclusion

- Assembly status: unsolved
- Benchmarks: needed
- Tools presented here: KNOT, Badread

Thank you! Any questions? (for real now)

Acknowledgements: Pierre Marijon, Jean-Stéphane Varré, Antoine Limasset, Camille Marchet, Brian Bushnell, Sergey Nurk, Marco Previtali, Paul Medvedev, Shaun Jackman, Guillaume Rizk, Ryan Wick, Tablet software, Adam Phillippy