A little tour of assembly methods

Antoine Limasset & Camille Marchet
CRIStAL, Université de Lille, CNRS, France
Evomics ’22 – Český Krumlov

@CamilleMrcht camille.marchet@univ-lille.fr
@BQPMalfoy antoine.limasset@univ-lille.fr
• Content of this course

• How to reconstruct a genome with sequencing data?
• What are the main challenges?
• Which solutions have been proposed?

• Easter egg Warning: 2 assembler names hidden in the slides

genome size: $\sim 32$ gigabases
Accessing a genome

From www.genome.gov/genetics-glossary/acgt
• Reads are words from the genome
• Reads are **shuffled** words from the genome
• Genome assembly task
Using read overlaps
Genome sequencing: coverage
Genome sequencing: coverage
Genome sequencing: coverage
Genome sequencing: coverage
Genome sequencing: coverage

NB: 30-100X are often required for assembly projects
How to assemble

Reads:
1: ATCGGTATCG
2: GG TATCGTTA
3: ATCGTTACGG
4: GTTACCGGTAT
5: ACGGTATACC

1: ATCGGTATCG
2: GG TATCGTTA

Overlap length: 7

1: ATCGGTATCG
3: ATCGTTACGG

Overlap length: 4

1: ATCGGTATCG
4: GTTACCGGTAT

Overlap length: 1

1: ATCGGTATCG
5: ACGGTATACC

No Overlap
- Assembly idea number 1: assemble the longest overlaps

Reads:
1: ATCGGTATCG
2: GGTATCGTTA
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATACC

Best overlaps: brought by each read
1: ATCGGTATCG
2: GGTATCGTTA
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATACC

Output "genome":
ATCGGTATCG + TTA + ATCGTTACGG + TAT + ACC
Your time to shine!

Let assemble this genome!

Your read set:

1: ATTTACGGGT
2: TTACGGGTGG
3: ACGGGTCCCTT
4: GTCCTTTTCCTT
5: TTTCTTACCGG

For each read:
  Find the best overlap (length>5)
  Merge the two reads
The Greedy solution

The best overlaps:
ATTTACGGGT
  TTACGGGTTGG
ACGGGTCCCTT
    GTCCCTTTCTT
        TTTCTTACGG

Output “genome”
ATTTACGGGTGG
ACGGGTCCCTTTCTTACGG
The actual solution

The actual genome:

ATTTACGGGTCCCTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT
ACGGGTCCCTT
GTCCTTTCTT
TTTCTTACGG
TTTACGGGTGG

longest overlap we found

ATTTACGGGT
TTTACGGGTGG

6

8
What happened?

The actual genome:

ATTTACGGGTCCCTTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT
ACGGGTCCCTT
6 GTCTTTTCTT
TTTCTTACGG
TTACGGGTGG

longest overlap we found

ATTTACGGGT
TTACGGGTGG

8

not in the genome

ACGGGTCCCTTTCTTACGG
not in the genome
Do we expect many repeats?

Probability to have NO repeated word of size 31 in a 5 megabases genome

Input interpretation:

\[
\left( \frac{4^{31} - 1}{4^{31}} \right)^{1/2} \left( 5 \times 10^6 \left( 5 \times 10^6 - 1 \right) \right)
\]

Decimal approximation:

0.999997289498784302383172055421363836712023171938932024106...

From en.wikipedia.org/wiki/Birthday_problem
The burden of assembly: genomic repeats

### Amount of repeats larger than a given size in *E. coli* genome

- 15: 44,994
- 21: 1,169
- 31: 559
- 41: 323
- 51: 225
- 61: 192

Genomic repeats are NOT random events
• Greedy assemblers

• Simple and efficient scheme
• Rely on local best choice (greedy)
• May create errors because of local choices when there are repeats
History from the last century

"Oh man. We have to find something better than this greedy assembly."

The genome assembly consortium, circa 1997.
- Graph representation

A node is a sequence

an arc oriented between a source node and a sink node

an arc means there is an overlap between the end of the source node and the beginning of the sink node
Assembly idea number 2: consider all overlaps

Genome:
AGTTACGGGTCCCTTTCTTGACGGGTGG

Overlap graph:

Read=node
Overlap=edge
Greedy solution

Genome:
\[
\text{ATTTACGGGTCCCTTCTTACGGGTGG}
\]

Overlap graph:

Greedy assembly output:
\[
\text{ATTTACGGGTGGACGGGTCCCTTCTTACGG}
\]
One piece solution

**Genome:**

```
ATTTACGGGTCCCTTCTTACGGGTGG
```

**Overlap graph:**

```
ATTT ACGGGT

6

Read=node

Overlap=edge

Overlap graph output:

ATTTACGGGTCCCTTCTTACGGGTGG
```
Multiple repeats

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTAAGT
AAGTATTT
ATTTGTT
TGTTGTC
TGTCTTTA

Overlap graph:

GCTGATTT ATTTGTAT GTATTGTC TGTAAGT AAGTATTT ATTTGTT TGTTGTC TGTCTTTA
First solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGCAAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

Possible assemblies:
GCTGATTT GTATTGTAT GTATTGTC TGCAAGT AAGTATTT ATTTTGTT TGTTTGTC TGTCTTTA

GCTGATTT ATTTGTAT GTATTGTC TGCAAGT AAGTATTT ATTTTGTT TGTTTGTC TGTCTTTA
Second solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAGT
AAGTATTT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:

Possible assemblies:
GCTGATTT ATTTGTAT GTATTGTC TGTCAGT AAGTATTT ATTTTGTT TGTTTGTC TGTCTTTA

Those two solutions are indistinguishable
Parsimonious solution: do not assemble

Possible assemblies:

Possible assemblies:

Genome pieces:

Repeats lead to the fragmentation of the assembly
Missing information also fragments the assembly
Assembly concession number 1: output fragments

In the real world, assemblers often provide pieces of genomes rather than complete ones.
- Overlap graph prerequisite: all overlaps
Overlap graph burden: number of reads

\[ n(n - 1)/2 = \mathcal{O}(n^2) \] possible overlaps for \( n \) reads

Linear: 2X data 2X time
Quadratic: 2X data 4X time
Overlap graph burden: number of reads

\[ n(n - 1)/2 = \mathcal{O}(n^2) \] possible overlaps for \( n \) reads

<table>
<thead>
<tr>
<th># Reads</th>
<th># Overlaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>499,500</td>
</tr>
<tr>
<td>10,000</td>
<td>50 million</td>
</tr>
<tr>
<td>100,000</td>
<td>5 billion</td>
</tr>
<tr>
<td>1 million</td>
<td>500 billion</td>
</tr>
<tr>
<td>10 million</td>
<td>50 trillion...</td>
</tr>
</tbody>
</table>

Most overlaps are too small to be considered...

The overlap computation is not linear

Talking about CPU years on large genomes...
Overlap graphs in a nutshell

- Graphs of overlaps between the reads
- Can provide a global solution for assembly
- Can be difficult in real cases because it requires a lot of computation (overlaps)

*S. cerevisiae, D. melanogaster*, human could be assembled using overlap graphs approaches (Celera (Myers et al. 2000), SGA (Simpson & Durbin 2011), ...)
● Fast forward

It's all about words

Genome Assembly Campus in the 2000's
• Assembly idea number 3: Focus on genome words

Let's select a word from the genome:

AGATACAGCCATGACCGTAGCATGCTAACTGTGACGGCATTAC

in the genome, after **TAGCAT**

only **AGCATG** appears
● Genome words / read words

In real cases we don't have the genome

AGATACAGCCATGACCGTAGCATGCTAACTGTTGA CGGCA T TAC

in the genome, after TAGCAT

only AGCATG appears

but we have the reads

ATGACCGTAGCATGCT
ATGACCGTAGCATGCT
GACCGTAGCATGCTAA

in the reads, after TAGCAT

only AGCATG appears
Reconstitute larger genomic words

extract all $k$-mers ($k = 7$):

AGATACAGCCATGACCGTAGCATGCTAAACTGTGACGGCATTAC

AGATACA
  GATACAG
    ATACAGC
      TACAGCC
        ACAGCCA
          CAGCCAT
            AGCCATG

AGATACA + G + C + C + A + T + G

AGATACAGCCATG a sequence from the genome
The de Bruijn graph

Read

AGATA\text{ACAGCCA}

De Bruijn graph

Kmer=node

\text{AGATA\cancel{CA}A} \rightarrow \text{GATA\cancel{CAG}} \rightarrow \text{\cancel{ATA}\cancel{CAGC}} \rightarrow \text{TACAG\cancel{CC}} \rightarrow \text{\cancel{ACAG\cancel{CCA}}}

k-1 overlap=edge

\text{AGATA\cancel{CA}A} + \text{G} + \text{C} + \text{C} + \text{A} = \text{AGATA\cancel{CA}CAGCCA}
• de Bruijn graph assembly

Overlapping reads
AGATACAGCCA
TACAGCCATGG

De Bruijn graph
AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG → GCCATGG

Overlap

Resulting sequence
AGATAČAGCCATGG
de Bruijn graph time!

Reads

GCTATGGGT
TACAGCCATG
AGCATTGGGT
GCCTTGGGT
AGATACAGCCA
ACAGCCATGG
GATACAGCCATG
CATGGTTAAA
ACAGCCATGG
GATACAGCCATG
CATGGTTAAA
CAGCCATGGGT

Hint: Use 7-mers
Solution

Overlapping reads

AGATACAGCC
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
ACAGCCATGGG
CAGCCATGGGG
AGCCATGGGGT
AGCCATGGGTT
GCCATGGGGTT
GCCATGGGGTT
CATGGGGTTTAA
CATGGGGTTTAA

De Bruijn graph

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG
GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGGTTTA → GGTTTAA

Resulting sequence

AGATACAGCCATGGGTTTAA
de Bruijn graph versus overlap graph

reads

\[ ...\text{AGCCATG}... \]
\[ ...\text{AGATACA}... \]
\[ ...\text{AGATACA}... \]

words from the reads

AGATACA
GATACAG TACAGCC AGCCATG
ATACAGC ACAGCCA CAGCCAT
...

word graph (de Bruijn graph)

\[ \text{AGATACA} \rightarrow \text{GATACAG} \rightarrow \text{ATACAGC} \rightarrow \text{TACAGCC} \rightarrow \text{ACAGCCA} \rightarrow \text{CAGCCAT} \rightarrow \text{AGCCATG} \]

Overlap graph from the reads

\[ ...\text{AGATACA}... \]
\[ ...\text{AGATACA}... \]
\[ ...\text{AGATACA}... \]
\[ ...\text{AGATACA}... \]
de Bruijn graphs abstract redundancy

Overlapping reads
AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
ACAGCCATGGG
CAGCCATGGGT
AGCCATGGGTTT
GCCATGGGTTT
GCCATGGGTTT
CATGGGTTTTAA
CATGGGTTTTAA

62 (non distinct) 7-mers in the reads

De Bruijn graph

14 distinct 7-mers in the De Bruijn graph

AGATACA → GATA → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG

GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGGTTTA → GTTTTAA
• de Bruijn graphs only rely on \( k - 1 \) overlaps

Overlapping reads

\[
\begin{align*}
AGATACAGCCA \\
GATACAGCCATG \\
GATACAGCCATG \\
TACAGCCATGG \\
ACAGCCATGGG \\
CAGCCATGGGT \\
GCCATGGGTT \\
CATGGGTTT \\
TGGGTTTA \\
GTTTTAA
\end{align*}
\]

De Bruijn graph overlap length: 6

\[
\begin{align*}
AGATACA & \rightarrow GATACAG & \rightarrow ATACAGC & \rightarrow TACAGCC & \rightarrow ACAGCCA & \rightarrow CAGCCAT & \rightarrow AGCCATG \\
GCCATGG & \rightarrow CCATGGG & \rightarrow CATGGGT & \rightarrow ATGGGTT & \rightarrow TGGGTTT & \rightarrow GGGTTTA & \rightarrow GTTTAA
\end{align*}
\]
- de Bruijn graphs limitation

**Fixed overlaps**

\[ \text{CAGGA} \rightarrow \text{AGGAC} \rightarrow \text{GGACT} \rightarrow X \rightarrow \text{ACTTA} \rightarrow \text{CTTAC} \]

- \text{GACTT} missing
- \text{not k-1 overlap}
de Bruijn graphs limitation

Repeats...

...TACAGGACTTA... ...TATAGGACTGA...

...→ TACAG → ACAGG → CAGGA → AGGAC → GGACT → GACTG → ACTGA → ...

...→ TATAG → ATAGG → TAGGA → ...

each k-mer appears only once in a de Bruijn graph
de Bruijn graph limitation

...TACAGGACTTA... ...TATAGGACTGA...

... → TACAG → ACAGG → CAGGA → AGGAC → GGACT → GACTG → ACTGA → ...

... → TATAG → ATAGG → TAGGA → GACTT → ACTTA → ...

 genome pieces

...TATAGGA AGGACT

GACTGA...

...TACAGGA GACTTA...
On the representation of de Bruijn graphs

De Bruijn graph:

Compacted De Bruijn graph:

Graphical representation (.gfa plot using Bandage):
• de Bruijn graph on a real dataset
de Bruijn graph on a real dataset ZOOMED IN
Sequencing errors

Genome:
```
ATCGGTATCGTTACGGTATACC
```

Reads:
```
A TCGCTATCG
GGTTTCGTTA
ATCGATACGG
TCGCTA
GGTTTC
ATCGAT
...
```

Are not genomic kmers...

Antoine Limasset & Camille Marchet
A little tour of assembly methods
Erroneous $k$-mers vs genomic $k$-mers

Genome:

\[ \text{TAAGAAAGCTCTGAATCAACGGACTGCGACA} \]

Reads:

\[
\begin{align*}
\text{TAAGAAAGCTCTGAATCA} & & \text{TCTGAATCA} \\
\text{AAGAAAGCTCTAAATCAAC} & & \text{AAGAAAGCTCTGAATCAACG} \\
\text{AGAAAGCTCTGAATCAACG} & & \text{...} \\
\text{GAAAGCTCTGAATCAACGG} & & \text{TCTGAATCAACGGACTGCGA} \\
\text{AAAGCTCTGAATCAACGGAC} & & \text{TCTGAATCAACGGACTGCGACA} \\
\text{AAGCTCTGAATCAACGGACT} & & \text{TCTGAATCAACGGACTGCGA} \\
\text{AGCTCTGAATCAACGGACTG} & & \text{TCTGAATCAACGGACTGCGA} \\
\text{GCTCTGAATCAACGGACTG} & & \text{TCTGAATCAACGGACTGCGA} \\
\text{CTCTGAATCAACGGACTGCG} & & \text{TCTGAATCAACGGACTGCGA} \\
\end{align*}
\]

9 times TCTGAAT
1 time TCTAAAT
6 times CAACGGA
1 time CAACGGT

Erroneous $k$-mers are seen less than genomic ones
- K-mer histogram
• Removing unique $k$-mers
- Removing $k$-mers seen less than 3 times
• Removing $k$-mers seen less than 4 times
Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads CAGGA

AGGAC GTAC

AGGACTTAC

GGACCTTACT

CAGGA → AGGAC → GGACG → GACGT → ACGTA → CGTAC → TTACT
Errors in de Bruijn graphs

...TAGAGGACTTACTGA... genome

reads

CAGGACTTA
AGGACTGTAC
AGGACTTTAC
GGACTTTACT

sequencing error

tip

CAGGA → AGGAC

GGACG → GACGT → ACGTA → CGTAC

GGACT → GACTT → ACTTA → CTTAC

TTACT
• Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads

ACAGGACTTA
CAGGAATTAC
CAGGACTTAC
AGGACTTACT

sequencing error

bubble

ACAGG
GGAAT
GGACT
AGGAA
AGGAC

CAGGA
GGAAT
GGACT
AA
TTA
ATTAC
TTACT
(Almost assembled phage !)
de Bruijn graphs in a nutshell

- Graph of words of size \( k \), \( k-1 \) overlaps
- Collapses identical \( k \)-mers
- Very successful, have replaced the overlap graphs with high throughput sequencing data
- Still outputs fragments of the genome

White spruce, 20 gigabases
Multiple $k$ assembly

Most de Bruijn graph assemblers can now perform several assemblies with different $k$-mer sizes to produce an improved "super" assembly.

**Exercice**
Build DBG with $k=5$ and $k=7$ from those reads:
AAAATCGATCTC
TCTCATCGAATT

TCTCATCGAATT
Multiple $k$ assembly

Most de Bruijn graph assemblers can now perform several assemblies with different $k$-mer sizes to produce an improved "super" assembly.
Multiple $k$ assembly

Most de Bruijn graph assemblers can now perform several assemblies with different $k$-mer sizes to produce an improved "super" assembly.
• de Bruijn graph on an eukaryota
- Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGGCCGCCAAAATGTGCTGGCCGGAC

♂ GGATGAAACAGCCGGTCAGGACACCCCTCTGAGGCCGGAAGGAAAATGTGCTGACCGGAC
Two or more genomes per individual

♀ GGATGAAAC\textcolor{red}{T}GCCGGTCAGG\textcolor{red}{T}CACCCCTCTGAGCCGCC\textcolor{red}{C}AAAATGCTGCTGGCCGGAC

♂ GGATGAAAC\textcolor{red}{A}GCCGGTCAGG\textcolor{red}{A}CACCCCTCTGAGCCGGC\textcolor{red}{G}AAAATGCTGCTG\textcolor{red}{A}CCGGAC

Assembly:
GGATGAAAC\textcolor{red}{T}GCCGGTCAGG\textcolor{red}{T}CACCCCTCTGAGCCGCC\textcolor{red}{C}AAAATGCTGCTGGCCGGAC
GGATGAAAC\textcolor{red}{A}GCCGGTCAGG\textcolor{red}{A}CACCCCTCTGAGCCGGC\textcolor{red}{G}AAAATGCTGCTG\textcolor{red}{A}CCGGAC
Assembly concession number 2: collapse variability

♀ GGATGAAAC T GCCGGTCAGGT CACCCCTCTGAGCCGCC GAAAAATGTGCTGGCCGGAC
♂ GGATGAAAC AGCCGGTCAGGA CACCCCTCTGAGCCGG GAAAAATGTGCTGACCGGAC

Assembly:
GGATGAAACT T GCCGGTCAGGA CACCCCTCTGAGCCGG GAAAAATGTGCTGGCCGGAC

Reads:
GATGAAACT TG
ATGAAAC AGC
TGAAAC AGCCG
GAAACT TGCCGG
AAACT TGCCGGT
AACAGCCGGTC
ACAGCCGGTCA
CTGCCGGTCAG
• Paralog genes/repeats
Paralog genes/repeats in graph

Sequencing error

Heterozygous variant

Quasi repeats

Antoine Limasset & Camille Marchet
A little tour of assembly methods 73 / 140
An assembler is a set of heuristics

Graph cleaning heuristics

- Nodes coverage
- Graph local/global topology
- Reads that can be mapped on nodes
- Estimated coverage/genome size
- ...
• An assembly is a model

1. Assemblies contain errors
2. Different tools can produce very similar assemblies
3. A single tool can produce very different assemblies with small changes of parameters(!)

From github.com/rrwick/Long-read-assembler-comparison
What do we do post-assembly?

1. Assess its quality
2. Improve it
3. Use it!
• Visualize assembly

Bandage tool can visualize assembly graphs (GFA)

From rwick.github.io/Bandage
• Visualize assembly

Bandage tool can visualize assembly graphs (GFA)
Assembly continuity

N50

N50 can be described as a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value.

Example: 1 Mbp genome

50%

1000

300 100 45 45 30 20 15 15 10 . . .
### Assembly continuity

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N50</strong></td>
<td>N50 can be described as a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value.</td>
</tr>
<tr>
<td><strong>N75</strong></td>
<td>N75 is the same statistic for 75% of the assembly.</td>
</tr>
<tr>
<td><strong>NGA50</strong></td>
<td>Similar to the N50 but only takes into account contigs/scaffolds that can be aligned on the reference genome and consider 50% of the genome size instead of the assembly size.</td>
</tr>
</tbody>
</table>

Antoine Limasset & Camille Marchet

A little tour of assembly methods
Evaluate assembly

Contigs can be mapped and compared to a reference/closely related genome

From quast.bioinf.spbau.ru/manual.html
- Misassemblies

- Contig
- Reference
- Chromosome 1  Chromosome 2

- Relocation
- Inversion
- Translocation

Chr 1  Chr 2  Chr 1  Chr 2  Chr 1  Chr 2
K-mer spectrum visualization with KAT
K-mer spectrum visualization with KAT

Assembly duplication histogram

Distinct Kmer Count

0x
1x
2x
3x
4x
5x

2e+07
1.5e+07
1e+07
5e+06
0 0 10 20 30 40 50 60 70 80 90 100 110 120 130 140
K-mer spectrum visualization with KAT

Assembly duplication histogram

Distinct Kmer Count

0x
1x
2x
3x
4x
5x
K-mer spectrum visualization with KAT

![K-mer comparison plot](https://kat.readthedocs.io/en/latest/)

Antoine Limasset & Camille Marchet

A little tour of assembly methods
Scaffolding

Softwares can improve the assembly continuity by using other kinds of information.

From ”Modern technologies and algorithms for scaffolding assembled genomes” Plos Computational Biology
The end

...of the theoretical part
Intermission

TBdBG or not TBdBG
that is the question
Sanger

- Medium reads $\approx 1000bp$
- Very low error rate $\approx 0.01\%$
- Low throughput (up to billion of reads per run)
- Costly ($500/Mb$)

No longer used for assembly
Second generation sequencing

- Short reads \( \approx 150 \text{bp} \)
- Low error rate \(< 1\%\)
- High throughput (up to billion of reads per run)
- Cheap ($0.50/\text{Mb})
- GC bias

Mainly assembled using de Bruijn graphs
State-of-the-art

Well performing assemblers
- SPAdes [Bankevich 2012]
- Megahit [Li 2015]
- IDBA [Peng 2012]

Other notable assemblers
- SGA [Simpson 2012]
- Discovar denovo [Weisenfeld 2014]
- Abyss [Simpson 2009]
Third generation sequencing

- Long reads $\approx 10 - 100\text{kbp}$
- High error rate $\approx 10\%$
- High throughput (up to millions of reads per run)
## Nanopore VS Pacbio

### Nanopore
- Portable
- Ultra long reads (100kbases, some reads reach the megabase level)
- Mostly deletions

### Pacbio
- More mature
- HiFi reads (99.9% identity)
- Mostly insertions
Repeats spanning

Genome:
GGTAATGGTTTTTTGGTGCTAAATGCGTTTTTTCATGATGTCGTAATTTTTTATCTG

Reads:
GGTAATG TTTTTT GTGCTAAT GTTTTTTT ATGGATG TTTTTTTA
ATGGTTT AATGCGTT ATGTCGT TTTATCTGG
TTTGGTG TTTTCATG CGTAATT

Contexts of the repeat:

...ATGG

???TTTTTT???

...TGCG

...GTAA

ATCT...

GGTG...

CATG...
Repeats spanning

Genome:

```
GGTAATGGTTTTTGGTCTAATGCGTTTTTTCATGGATGCTGAAATTTTTTATCT
```

Reads:

```
GGTAATGTTTTTGTGCTAATGTGGATGTTTTTTCATGGATGCTGAAATTTTTTATCT
```

Long reads:

```
TGGTTTTTGGT TGGCTTTTTTCATGAAATTTTTTATCT
```

Contexts of the repeats:

```
...ATGG → GGTG...
...TGCG → CATG...
...GTAA → ATCT...
```
• Read length matter

Read size=21
Read length matter

Read size=31
Read length matter

Read size = 63
Read length matter

Read size=255
Read length matter

Read size=500
- Read length matter

Read size = 1000
Read length matter

Read size=2000
Great hope for assembly

From "One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly" Current Opinion in Microbiology 2015
Long reads killed the assembly star

Laura Landweber @LandweberLab · Jan 2
Our newest version of Oxytricha's somatic genome is out (rdcu.be/bZNfC) and has 18,617 distinct chromosomes. That's 2000 more than we previously published in doi.org/10.1371/journale. PacBio captured most chromosomes in single reads: Genome sequence, No assembly required
Great hope for assembly

From "Chromosome-scale assemblies of plant genomes using nanopore long reads and optical maps" Nature Plants 2018
Which assembly strategy is best suited?

- Long reads $\approx 10kbp$
- High error rate $\approx 10\%$
- High throughput (up to millions of reads per run)

Based on long reads properties, which assembly solution would you choose and why?

Vote!
- Greedy
- Overlap graph
- de Bruijn graph
Long reads for assembly: de Bruijn graph?

- Many errors in k-mers:
  - Noisy graph
  - Disconnected genome reads
**Long reads for assembly: overlap graph?**

Supposed to be super expensive!

Average coverage: 10
Read length: 10
Average overlap: 9
Read number: 100

Average coverage: 10
Read length: 30
Average overlap: 27
Read number: 33
• Longer reads, better overlaps
  • Less reads for the same coverage
  • Larger overlaps

5Mb bacteria example with 100X coverage

<table>
<thead>
<tr>
<th>Short reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 5 million 100bp reads</td>
</tr>
<tr>
<td>• 99 bp average overlap</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Long reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 50,000 10kbp reads</td>
</tr>
<tr>
<td>• 9,900 bp average overlap</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Very long reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 5,000 100kbp reads</td>
</tr>
<tr>
<td>• 99,000 bp average overlap</td>
</tr>
</tbody>
</table>
Are large overlaps hard to compute?

Aligning very long and highly erroneous regions is expected to be expensive, as alignment is quadratic $\approx \mathcal{O}(n^2)$!
"Anchor chaining" in overlap graph

For long reads: typically Minimap’s [Li 2016] job

"Anchor chaining": find common chains of anchors (k-mers) in the same order in 2 sequences (can be linear in practice in most cases)
Long reads for assembly: overlap graphs

- Genome reads
- Compute overlaps
- Overlap graph

Antoine Limasset & Camille Marchet

A little tour of assembly methods 113 / 140
Sequencing errors

Genome:

\[
\text{ATCGGTATCGTTACGGTATACC}
\]

Reads:

\[
\text{ATCGCTATCG} \quad \text{(Substitution)}
\]
\[
\text{GGTATCGTCTA} \quad \text{(Insertion)}
\]
\[
\text{ATGTTA} \quad \text{ATCGG} \quad \text{(Deletion)}
\]

Insertion and deletion made calling almost impossible
Using coverage to remove noise: Consensus

Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGGTATCCAGAATTTGTCACTT

Reads:
AAAGAAAGCATCTGAATCAACGGACTTCGAG
GAAAGCTCTCAACCAACGGACTGCGACATT
ACCTCTCAAGCAACGGACTGCGACAAAAAG
TCTGAATCAACCGGACTGCGTCAAAAAGTGC
GAATCAACCGGACTGCGACAGTTTGTGGTGG
TCAACGCAGTGGGACACGAAAAATGGG
GACTGCCAATGGGTGGATCCCA
AACGACATCGGATATCCAGAGTTG
GGACCATGGAAGTTGTGCAGG
TAAAAGATATGCGATTCCAGAGTTG

Consensus:
AAAGATAGCTCTGAATCAACGGACTGCGACAAAGTGGTGGTATCCAGAATTTTTCAGTT

1/1              4/7    9/10   6/11                  3/4
Exercise: Perform a consensus

Erroneous reads:
TAAGAAAGCCTCTGAATCAAACCGGTACTGCGA
GAAAGCTTTGAATCAAACCGGACTGCGACAA
AGCTCTGAATCAAACCGGACTGCGACAATAA

Contig to polish:
TAAGAAAGCTTTGAATCAAACCGGGAATGGCGACAAATAA
Exercise: Perform a consensus - solution

Correct contig:
TAAGAAAGCTCTTAATCAA-CGGACTG-CGACAATAA

Aligned reads:
TAAGAAAGCTCTTAATCAAACGGACTGTCGA
AAAGCT- TAATCAA-CGGACTG-CGACAAT
GCTCTTAATCAA-CGG-CTG-CGACAATAA
Consensus during assembly

... overlap graph ...

consensus
• Consensus after assembly: polishing

1. align reads
2. compute consensus
3. correct assembly with consensus

contig
- Consensus after assembly: polishing

1. align reads
2. compute consensus
3. correct assembly with consensus

contig

1. align reads
2. compute consensus
3. correct assembly with consensus

AAAAA
Homopolymers are hard to read
Polishing using accurate reads

1. align short reads
2. compute consensus
3. correct assembly with consensus
Systematics errors

Polishing with Illumina data can improve the final error rate

**A. Single-tool short-read polishing**

<table>
<thead>
<tr>
<th>ALE change:</th>
<th>0</th>
<th>110696</th>
<th>113366</th>
<th>87707</th>
<th>113056</th>
<th>113061</th>
<th>115623</th>
<th>82446</th>
</tr>
</thead>
<tbody>
<tr>
<td>total distance:</td>
<td>7635</td>
<td>212</td>
<td>74</td>
<td>2519</td>
<td>1775</td>
<td>128</td>
<td>28</td>
<td>1867</td>
</tr>
</tbody>
</table>

From Polypolish: Short-read polishing of long-read bacterial genome assemblies
● Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometime the software fail

From github.com/rrwick/Long-read-assembler-comparison
• Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the data cannot solve the problem

- Very large repeated region
- Low local coverage
- Chimeric/noisy reads
20 years later
Telomere-to-Telomere consortium

Has produced in 2021 a complete human genome with one contig per chromosomes!

- 30x PacBio HiFi
- 120x coverage of Oxford Nanopore (ultra long reads)
- 70x PacBio CLR
- 10X Genomics, BioNano DLS and Arima Genomics HiC
- 100 authors from 50 labs
• Long reads assemblers

<table>
<thead>
<tr>
<th>Best performing assemblers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Flye (Repeat graph) [Kolmogorov et al 2019]</td>
</tr>
<tr>
<td>• Raven (OLC) [Vaser et al 2021]</td>
</tr>
<tr>
<td>• NECAT/MECAT(OLC) [Xiao et al 2017]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other notable assemblers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Canu (Greedy) [Koren et al 2017]</td>
</tr>
<tr>
<td>• Shasta (OLC) [Saffin et al 2020]</td>
</tr>
<tr>
<td>• Redbean (fuzzy de Bruijn graph) [Ruan 2019]</td>
</tr>
<tr>
<td>• ...</td>
</tr>
</tbody>
</table>
Long read assembly summary

- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs
Consensus during sequencing

HiFi data

Stands for "High Fidelity"
Very low error rates $\approx 0.1\%$ $0.01\%$
Almost only homopolymer errors remain
HiFi Assembly

With almost error-less long reads we have several promising improvements ahead:

- Use de Bruijn graph (more efficient data structures)
- Assemble large genomes very fast
- Perform diploid assembly
de Bruijn graph Assembly

Using $K=500$ and $K=5000$ de Bruijn graphs to assemble
Very fast genome assembly

Human genome assembled within 2 hours (Peregrine assembler) and 10 minutes (RMBG assembler)
Diploid assembly
Ongoing progress

Errors in Nanopore sequencing data are rapidly diminishing

Q20 chemistry achieved modal accuracy > 99%
High fidelity nanopore incoming?

Nanopore duplex reads could deliver long and precise reads in the future
The end

That's all Folks!
- Take home messages

### Ultra fast summary
- Short reads: de Bruijn graphs / Long reads: Overlap graph
- Repeats are the core issue
- Output fragments of genomes (contigs)
- Several parameters and heuristics used in practice
On going work

Assembly Challenges

- Reconstruct haplotypes
- Scaling on large genomes
- Robustness to noisy data
- Repetitive regions
The end

i trust you to figure out your own genome

Traduire le Tweet
3:01 AM · 14 déc. 2019 · Twitter for iPhone

37 Retweets 267 J'aime