A little tour of assembly methods

Antoine Limasset & Camille Marchet
CRIStAL, Université de Lille, CNRS, France
Evomics ’23 – Český Krumlov

@Camillemrcht    camille.marchet@univ-lille.fr
@BQPMalfoy       antoine.limasset@univ-lille.fr
Antoine Limasset

Things he likes

Research

Random facts
- used to look like Drago Malfoy
- Amazon integrated one of his methods to their Alexa
- has done his PhD defense slides deck on an emergency lane
- has directed a movie about assembly
- will solve advanced data structure problems in his sleep

Treasure hunt: find Antoine's fav animal in one of the slides

2020, 2022, 2023

Evomics

Places

According to MidJourney

current

Treasure hunt: find Antoine's fav animal in one of the slides
Camille Marchet

Can name every bird

Focus on transcriptomic,
Genomic is too easy

Can know if a restaurant is good
by the colors of the chairs

Deadline is 4 day away?
Let’s submit another paper

Learn novel language during a plane trip

Manuscript rejected?
Let submit to a better journal

Parental Leave?
Goes to London and Krumlov
the same week with two child

Read half of a CS book in her life
Develop state of the art data structures

Tries new recipes every day
100% success rate
Content of this course

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

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

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

- 4X
- 2X
- 1X
Genome sequencing: coverage
Genome sequencing: coverage
Genome sequencing: coverage
Genome sequencing: coverage
Genome sequencing: coverage

30-100X are often required for assembly projects
Which overlap?

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

1: ATCGGTATCG
2: GGTATCGTTA
Overlap length: 7

1: ATCGGTATCG
3: ATCGTTACGG
Overlap length: 4

1: ATCGGTATCG
4: GTTACCGTAT
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: ACGGTATAACC

Best overlaps:
1: ATCGGTATCG
2: GGTATCGTTA
3: ATCGTTACGG
4: GTTACGGTAT
5: ACGGTATAACC

Output "genome": supplementary information brought by each read

ATCGGTATCG + TTA + CGG + TAT + ACC
Let assemble this genome!

Your read set:

1: ATTTACGGGT
2: TTACGGGTGG
3: ACGGGTCCCTT
4: GTCCCCCTT
5: TTTCTTACGG

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

The best overlaps:
ATTTCGCGGT
  TTACGGGTTGG
ACGGGTCCCCTT
    GTCCTTTTCTT
        TTTCTTACGG

Output “genome”
ATTTCGCGGTTGG
ACGGGTCCCCTTCTTACGG
The actual solution

The actual genome:
ATTTACGGGTCCCTTTCTTACGGGTTGG

How the reads should be ordered:
ATTTACGGGT
ACGGGTCCCTT
GTCCTTTCTT
TTTCTTACGG
TTACCGGTGG

longest overlap we found
6 ATTTACGGGT
TTACCGGTGG
8
What happened?

The actual genome:

ATTTACGGGTCTTTTCTTACGGGTGG

How the reads should be ordered:

ATTTACGGGT
ACGGGTCTTT
6
GTCCTTTTCTT
TTTCTTACGG
TTACGGGTGG

longest overlap we found

ATTTACGGGT
TTACGGGTGG

8

not in the genome

ACGGGTCTTTTCTTACGG

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: the human genome project while finding a successor to the greedy approach (according to MidJourney)
- 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:
\texttt{ATTTACGGGTCCCTTCTTACGGGTGG}

Overlap graph:
Greedy solution

Genome:
\texttt{ATTATCGGGTCCCTTCTTACGGGTGG}

Overlap graph:

\begin{figure}
\centering
\includegraphics[width=\textwidth]{overlap_graph.png}
\end{figure}

Read=\texttt{node}
\texttt{ATTATCGGGT}

Overlap=\texttt{edge}
\texttt{TTACGGGTGG}

Greedy assembly output:
\texttt{ATTATCGGGTGG}
\texttt{ACGGGTCCCTTCTTACGG}
One piece solution

Genome:
\[\text{ATTTACGGGTCCCTTTCTTACGGGTGG}\]

Overlap graph:

Overlap graph output:
\[\text{ATTTACGGGTCCCTTTCTTACGGGTGG}\]
Multiple repeats

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAGGT
AAGTATTG
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:
First solution

Reads:
GCTGATTT
ATTTGTAT
GTATTGTC
TGTCAGTT
AAGTATT
ATTTTGTT
TGTTTGTC
TGTCTTTA

Overlap graph:
Possible assemblies:
GCTGATTT GTATT GTATTGTC TGTCAAGT AAGTATTT ATTTTGTT TGTTTGTC TGTCTTTA

Possible assemblies:
GCTGATTT GTATT GTATTGTC TGTCAAGT 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:

- GCTGATTT → ATTTGTAT → GTATTGTC → TGICAGT → AAGTATT → ATTTTGTT → TGTTTGTC → TGTCCTTA
- GCTGATTT → ATTTTGTATTGTC → TGICAGT → AAGTATT → ATTTTGAT → GTATTGTC → TGTCCTTA

Genome pieces:

- GCTGATTT
- ATTTGTAT
- TGICAGT
- AAGTATT
- ATTTTGTT
- TGTTTGTC
- TGTCCTTA

Repeats lead to the fragmentation of the assembly
Missing information also fragments the assembly
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

\[ n(n - 1)/2 = O(n^2) \] 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>

The overlap computation is not linear
Talking about CPU years on large genomes... We have to be efficient here and focus on "relevant" overlaps
Overlap graph simplifications

- remove small overlaps
- remove node inclusions
- remove dominated overlaps
● 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), ...).
• History: the introduction of another graph structure for assembly, according to MidJourney
• Assembly **idea number 3: Focus on genome words**

A genome: a succession of words

```
AGATACAGCCATGACCGTGCTAAGCTAACTGTGACGGCATTTAC
GCATGC
CATGCT
```

Antoine Limasset & Camille Marchet

A little tour of assembly methods 42 / 160
• Find consecutive genome words in read words

genome

AGATACACGACATGACCCTAGCATGCTAACTGTGACGCGCATTAC

read

TGACCGTAGCATGCT

TGACCG

GACCGT

...  k-mers  GCATGCT

(k=6)  CATGCT

next nucleotide

 Antoine Limasset & Camille Marchet

A little tour of assembly methods  43 / 160
How to connect read words?

genome

AGATACAGCCATGACCGTAGCATGCTAACTGTGTA
GACGGCATTA

reads

TGACCGTAGCATGCT
GACCGTAGCATGCTA

k-mers
(k=6)

GCATGCT
CATGCT
ATGCTA

next nucleotide
Reconstitute larger genomic words

A little tour of assembly methods
The de Bruijn graph

Read

AGATACAGCCA

De Bruijn graph

Kmer=node

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA

k-1 overlap=edge

AGATACA + G + C + C + A

=AGATACAGCCA
de Bruijn graph assembly

Overlapping reads

Resulting sequence

AGATACAGCCATGG

De Bruijn graph

AGATACAGCCATGG
de Bruijn graph time!

Reads

```
GCCATGGGTTT
TACAGCCATGG
AGCCATGGGTT
GCCATGGGTTT
AGATACAGCCA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
ACAGCCATGGG
GATACAGCCATG
CATGGGTTTAA
CAGCCATGGGT
```
Solution

Overlapping reads

AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
ACAGCCATGGG
CAGCCATGGGT
AGCCATGGGTTT
GCCATGGGTTTT
GCCATGGGTTTT
CATGGGTTTTAA
CATGGGTTTTAA

De Bruijn graph

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG

GCCATGG → CATGGGTTTTAA

Resulting sequence
AGATACAGCCTGGGTTTTAA
Overlapping reads:

AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
ACAGCCATGGG
CAGCCATGGGT
AGCCATGGGT
GCCATGGGTTT
GCCATGGGTTT
CATGGGTTTTAA
CATGGGTTTTAA

62 (non distinct) 7-mers in the reads

De Bruijn graph:

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG

GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGGTTTA → GGGTTTA

14 distinct 7-mers in the De Bruijn graph

Antoine Limasset & Camille Marchet
• de Bruijn graphs only rely on $k - 1$ overlaps

Overlapping reads

AGATACAGCCA
GATACAGCCATG
GATACAGCCATG
TACAGCCATGG
ACAGCCATGGG
CAGCATGGGTT
GCCATGGGGTT
ATGGGGTT
TGGGTTT
GGTTTA

De Bruijn graph overlap length: 6

AGATACA → GATACAG → ATACAGC → TACAGCC → ACAGCCA → CAGCCAT → AGCCATG

GCCATGG → CCATGGG → CATGGGT → ATGGGTT → TGGGTTT → GGTTTA → GGTAAAA
• de Bruijn graphs limitation 1: Fixed overlaps

CAGGA → AGGAC → GGACT \[\times\] ACTTA → CTTAC

**GACTTT** missing
not k-1 overlap

GGACT and ACTTA overlap is only of size 3 !
• de Bruijn graphs limitation 2: Repeats

...TACAGGACCTTA... ...TATAGGACTGA...

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

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

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 → ...

...TATAGGA

GACTGA...

AGGACT

GACTTA...

 genome pieces
de Bruijn graph versus overlap graph

reads

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

words from the reads

\[ \text{AGATAAC} \]
\[ \text{GATAACAG} \]
\[ \text{TACAGCC} \]
\[ \text{AGCCATG} \]
\[ \text{ATACAGC} \]
\[ \text{ACAGCCA} \]
\[ \text{CAGCCAT} \]
\[ ... \]

word graph (de Bruijn graph)

\[ \text{AGATAAC} \rightarrow \text{GATAACAG} \rightarrow \text{ATACAGC} \rightarrow \text{TACAGCC} \rightarrow \text{ACAGCCA} \rightarrow \text{CAGCCAT} \rightarrow \text{AGCCATG} \]

Overlap graph from the reads

\[ \text{...AGATAAC...} \]
\[ \text{...AGATAAC...} \]
\[ \text{...AGATAAC...} \]
\[ \text{...AGATAAC...} \]
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:
ATCGGTATCGTTACGGTATAACC

Reads:
ATCGCTATCG
GGTTTCGTTA
ATCGATACGG
TCGCTA
GGTTTC
ATCGAT
...
Are not genomic kmers...

Antoine Limasset & Camille Marchet
Erroneous $k$-mers vs genomic $k$-mers

Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACA

Reads:
TAAGAAAGCTCTGAATCA
AAGAAAGCTCTAAATCAAC
AGAAAGCTCTGAATCAACG
GAAAGCTCTGAATCAACGGGA
AAAGCTCTGAATCAACGGAC
AGCTCTGAATCAACGGACT
AGCTCTGAATCAACGGACTG
GCTCTGAATCAACGGAC
CTCTGAATCAACGGACTGCG
TCTGAATCAACGGACTGCGA

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

Erroneous $k$-mers are seen less than genomic ones
K-mer histogram

K-mer comparison plot

Number of distinct k-mers

k-mer multiplicity

0x
1x
2x
3x
4x
5x
6x+
• 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

CAGGACTTA
AGGACGTAC
AGGACTTAC
GGACCTTACT

sequencing error

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

...TACAGGACTTACGTGA... genome

reads

CAGGACTTA
AGGACTGTA
AGGACTTAC
GGACTTACT

sequencing error

tip

CAGGA
AGGAC
GGACT
GACGT
GACTT
ACGTA
ACTTA
CGTAC
CTTAC
TTACT
Errors in de Bruijn graphs

...TACAGGACTTACTGA... genome

reads ACAGGACTTA
CAGGAAATTAC ← sequencing error
CAGGACTTAC
AGGACTTACT

bubble

ACAGG → CAGGA → AGGAA → GGAAT → GGAAT → GAATT → AATTA → ATTAC
AGGAC → GGACT → GGACT → GACCTT → ACTTA → CTTAC → 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
Multiple $k$ assembly

We are missing GATCTCA and ATCTCAT in the second graph. But they are present in the first graph!
Multiple $k$ assembly

compacted de Bruijn graph
$k=5$

AAAATCG → ATCGA → TCGATCTCATCG

k-mers $k=7$
from cdBG $k=5$

AAAATCG
AAATCGA
TCGATCT
GATCTCA
ATCTCAT
TCTCATC
CTCATCG

k-mers $k=7$
from the reads

AAAATCG
AAATCGA
AATCGAT
ATCGATC
TCGATCT
GATCTCA
ATCTCAT
TCTCATC
CTCATCG

AAAATCGATCTCATCGAATT

compacted de Bruijn graph
$k=7$

AAAATCGATCTCATCGAATT
de Bruijn graph on an eukaryota
- Two or more genomes per individual

♀ GGATGAAACTGCCGGTCAGGTCACCCCTCTGAGCCGGGAAAAATGTGCTGACCGGAC

♂ GGATGAAACTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAAATGTGCTGACCGGAC
Two or more genomes per individual

♀ GGATGAAACTGCCGTCAGGTCAACCTCTCTGAGCCGCCC AAAATGTGCTGCCGGAC

♂ GGATGAAACAGCCGTCAGGAACACCCCTCTGAGCGCGGAAAATGTGCTGACCGGAC

Assembly:
GGATGAAACTGCCGTCAGGTCAACCTCTCTGAGCCGCCCAAAATGTGCTGCACCGGAC

Antoine Limasset & Camille Marchet
Assembly concession number 2: collapse variability

♀ GGATGAAACGTGCCGGTCAGGTCACCCCTCTGAGCCGCCAAAATGTGCTGGCCGGAC
♂ GGATGAAACCAGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

Assembly:
GGATGAAACGTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC

Reads:
GGATGAAACGTGCCGGTCAGGACACCCCTCTGAGCCGGGAAAATGTGCTGACCGGAC
• Paralog genes/repeats
- Paralog genes/repeats in graph

Sequencing error

1X

100X
GGATGAAAAC

100X
AACAGCC

Heterozygous variant

50X

100X
AGGTCAC

100X
CACCCCTCTGAGCCG

100X

50X

100X
AGGACAC

100X

Quasi repeats

200X
GGATGAAAAC

200X
AGGTCAC

200X
CACCCCTCTGAGCCG

100X

100X
AACAGCC

100X
GCGGGTCAGG

100X
AGGACAC

Antoine Limasset & Camille Marchet

A little tour of assembly methods
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!

Two ways to polish an assembly according to Mid-Journey
Evaluate assembly according to a reference

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

From quast.bioinf.spbau.ru/manual.html
## Assembly statistics

<table>
<thead>
<tr>
<th>Alignment-based statistics</th>
<th>ABySS</th>
<th>MEGAHIT</th>
<th>SPAdes</th>
<th>Velvet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genome fraction (%)</td>
<td>98.661</td>
<td>98.424</td>
<td>98.113</td>
<td>97.997</td>
</tr>
<tr>
<td>Duplication ratio</td>
<td>1.043</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td># genomic features</td>
<td>4525 + 75 part</td>
<td>4511 + 64 part</td>
<td>4489 + 50 part</td>
<td>4486 + 56 part</td>
</tr>
<tr>
<td>Largest alignment</td>
<td>248 481</td>
<td>235 933</td>
<td>285 096</td>
<td>264 944</td>
</tr>
<tr>
<td>Total aligned length</td>
<td>4 776 214</td>
<td>4 568 317</td>
<td>4 553 809</td>
<td>4 550 150</td>
</tr>
<tr>
<td>NGA50</td>
<td>69 801</td>
<td>122 647</td>
<td>133 309</td>
<td>112 446</td>
</tr>
<tr>
<td>LGA50</td>
<td>21</td>
<td>14</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Misassemblies</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># misassemblies</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Misassembled contigs length</td>
<td>231 767</td>
<td>0</td>
<td>0</td>
<td>435 515</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Per base quality</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># mismatches per 100 kbp</td>
<td>2.09</td>
<td>2.69</td>
<td>1.03</td>
<td>3.19</td>
</tr>
<tr>
<td># indels per 100 kbp</td>
<td>0.57</td>
<td>1.31</td>
<td>0.29</td>
<td>1.98</td>
</tr>
<tr>
<td># N's per 100 kbp</td>
<td>24.59</td>
<td>0</td>
<td>17.55</td>
<td>94.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistics without reference</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># contigs</td>
<td>176</td>
<td>95</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Largest contig</td>
<td>248 481</td>
<td>235 933</td>
<td>285 196</td>
<td>264 944</td>
</tr>
<tr>
<td>Total length</td>
<td>4 777 853</td>
<td>4 571 292</td>
<td>4 557 363</td>
<td>4 552 266</td>
</tr>
<tr>
<td>Total length (&gt;= 1000 bp)</td>
<td>4 757 929</td>
<td>4 562 458</td>
<td>4 548 710</td>
<td>4 544 453</td>
</tr>
<tr>
<td>Total length (&gt;= 10000 bp)</td>
<td>4 562 801</td>
<td>4 478 614</td>
<td>4 466 223</td>
<td>4 475 223</td>
</tr>
<tr>
<td>Total length (&gt;= 50000 bp)</td>
<td>3 248 113</td>
<td>3 833 793</td>
<td>3 812 315</td>
<td>3 817 904</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BUSCO completeness</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete BUSCO (%)</td>
<td>98.65</td>
<td>98.65</td>
<td>98.65</td>
<td>98.65</td>
</tr>
<tr>
<td>Partial BUSCO (%)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted genes</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># predicted genes (unique)</td>
<td>3717</td>
<td>3595</td>
<td>3587</td>
<td>3576</td>
</tr>
</tbody>
</table>
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 ...
genome

ACGGATGATAGATTTGATACGA

GATTTGATAC

ACGGATGATA

TTTGATACGA

reads

ACGGATGATA

TTTGATACGA

concatenate the reads: super N50!

GATTTGATACACGGATGATATTTGATACGA
## Assembly continuity

<table>
<thead>
<tr>
<th>Statistic</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>
- Misassemblies

### Contig
- Reference: Chromosome 1 | Chromosome 2

### Relocation
- Chr 1 | Chr 2

### Inversion
- Chr 1 | Chr 2

### Translocation
- Chr 1 | Chr 2
• Visualize assembly

Bandage tool can visualize assembly graphs (GFA)

From rwick.github.io/Bandage
• Visualize assembly

Bandage tool can visualize assembly graphs (GFA)
**K-mer spectrum visualization with KAT**


Antoine Limasset & Camille Marchet

A little tour of assembly methods
**K-mer spectrum visualization with KAT**

Assembly duplication histogram

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

Distinct Kmer Count

0 5e+06 1e+07 1.5e+07 2e+07

0 20 40 60 80 100 120 140
K-mer spectrum visualization with KAT

Assembly duplication histogram

Distinct Kmer Count

Antoine Limasset & Camille Marchet
K-mer spectrum visualization with KAT

From https://kat.readthedocs.io/en/latest/

Antoine Limasset & Camille Marchet

A little tour of assembly methods 93 / 160
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

Antoine Limasset & Camille Marchet
• The end

...of the theoretical part (or is it?)
Intermission

The workshop’s team during the annual lungfish genome assembly session according to MidJourney
• Sanger

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

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
Which assembly strategy is best suited?

- Short reads \(\approx 100\text{bp}\)
- Low error rate below 1%
- High throughput (up to billions of reads per run)

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

Vote!
- Greedy
- Overlap graph
- de Bruijn graph
## Paradigm Breakdown

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Does not handle repeats</th>
<th>Scalable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Overlap graph</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>De Bruijn graph</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Greedy**
  - Does not handle repeats ×
  - Scalable ✓

- **Overlap graph**
  - Handles repeats smaller than reads size ✓
  - Extremely expensive to handle billion reads ×

- **De Bruijn graph**
  - Handle repeat smaller than $k \approx$ reads size ✓
  - Scalable ✓
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 $\approx 10 - 100kbp$
- High error rate (up to 10%)
- High throughput (up to millions of reads per run)
### Nanopore VS Pacbio

<table>
<thead>
<tr>
<th>Nanopore</th>
<th>Pacbio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portable</td>
<td>More mature</td>
</tr>
<tr>
<td>Ultra long reads (100k bases, some reads reach the megabase level)</td>
<td>HiFi reads (99.9% identity)</td>
</tr>
<tr>
<td>Mostly deletions</td>
<td>Mostly insertions</td>
</tr>
</tbody>
</table>
Repeats spanning

Genome:
GGTA\textcolor{red}{ATGG}TTTTTTT\textcolor{yellow}{GGTG}CTAA\textcolor{cyan}{TGC}GTTTTTT\textcolor{red}{CATG}\textcolor{cyan}{GATTG}CGTAA\textcolor{red}{TTTTTT}ATCTG

Reads:
GGTA\textcolor{red}{ATG}TTTTTT\textcolor{yellow}{GTG}CTAA\textcolor{cyan}{AATTG}GATGTTTTTTGATGCGTTTTTTCATGGATGTCGTAATTTTTTATCTG

Contexts of the repeat:

...\textcolor{red}{ATGG}...

??? TTTTTT ???

...\textcolor{red}{TGCG}...

...\textcolor{red}{GTAA}...

ATCT...

GGTG...

CATG...
Repeats spanning

Genome:
GGTAATGTTTTTGGTGCTAATGCCGTTTTTTCATGATGTCTGAAATTTTTATCTG

Reads:
GGTAATGTTTTTGTGCTAATGTTTTTTATGGATGTTTTTTATCTG

Long reads:
TGGTTTTTTGGTTGCTCCTTTTTTCATGATCTGAAATTTTTATCTG

Contexts of the repeats:

...ATGG → GGTG...
...TGCG → CATG...
...GTAA → ATCT...
• Read length matters
Read size=21
• Read length matters

Read size=31
Read length matters

Read size=63
• Read length matters

Read size=255
Read length matters

Read size=500
Read length matters

Read size = 1000
Read length matters

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/journa.... 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?

Most $k$-mers will contain at least an error and will be useless.
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>
<th>Long reads</th>
<th>Very long reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 million 100bp reads</td>
<td>50,000 10kbp reads</td>
<td>5,000 100kbp reads</td>
</tr>
<tr>
<td>99 bp average overlap</td>
<td>9,900 bp average overlap</td>
<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 $\approx O(n^2)$!
"Anchor chaining" in overlap graph

For long reads: typically Minimap2’s [Li 2018] 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
Sequencing errors

Genome:

\[
\text{ATCGGTA}\text{TCGTTACGGTATAACC}
\]

Reads:

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

High rate of insertions and deletions rendered genome annotation nearly impossible
Using coverage to remove noise: Consensus

Genome:
TAAGAAAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGATCCAGAATTGTCACTT

Reads:

Consensus:
AAAGATAGCTCTGAATCAACGGACTGCGACAATAAGTGGTGATCCAGAATTGTCACTT
Exercise: Perform a consensus

RS1: ACTTCGAACGT
RS2: TCGATCGTTT
RS3: GATCAGTTTAG
RS4: TCATTTCGTA
RS5: GTTTCGTCGG
REF: ACTCGAATGTTTTTCCTACG
Exercise: Perform a consensus - solution

RS1: ACTTCGA-AC-GT-
RS2: -T-CGA-TC-GTTT-
RS3: -GA-TCAGTTT-AG-
RS4: -TC-ATTT-TCGTA-
RS5: -GTTT-CGTCGC
REF: ACT-CGAAT--GTTTTCCTACG
CON: ACT-CGAATC-GTTT-CGTAACG
Consensus during assembly
• Consensus after assembly: polishing

1. align reads
2. compute consensus
3. correct assembly with consensus
Consensus after assembly: polishing

1. align reads
2. compute consensus
3. correct assembly with consensus
• 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
I know we said it was the end

Just one or two more graphs
● An overlap graph limitation. Swept under the carpet?
● An overlap graph limitation. Swept under the carpet?
An overlap graph limitation. Swept under the carpet?
• Long reads for assembly: assembly solved?

Assembly is not solved yet

Sometimes the software fails

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
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

### Best performing assemblers
- Flye (Repeat graph) [Kolmogorov et al 2019]
- Raven (OLC) [Vaser et al 2021]
- NECAT/MECAT (OLC) [Xiao et al 2017]

### Other notable assemblers
- Canu (Greedy) [Koren et al 2017]
- Shasta (OLC) [Saffin et al 2020]
- Redbean (fuzzy de Bruijn graph) [Ruan 2019]
- ...
FLYE

APPLIES EDMONDS’ ALGORITHM (EDMONDS, 1965) TO FIND A MAXIMUM WEIGHT MATCHING IN THE TRANSITION GRAPH AND USES THIS MATCHING FOR UNTANGLING THE CONTRACTED REPEAT GRAPH. AFTER ITERATIVE UNTANGLING OF EDGES IN THE CONTRACTED ASSEMBLYGRAPH (AND THE CORRESPONDING ITERATIVE REPEAT RESOLUTION IN THE ASSEMBLY GRAPH), THE ASSEMBLY GRAPH TYPICALLY CONTAINS ONLY LONG UNBRIDGED REPEAT EDGES THAT ARE NOT SPANNED BY ANY READS.
Repeat graph

a genome

highlighted repeated regions
Repeat graph

repeats extremities: graph's nodes
Repeat graph

sequences in-between: arcs
Repeat graph

collapse multiple arcs
Repeat graph
Long read assembly summary

- Overlap graphs with quick overlap computation
- Long reads can span repeats and improve assemblies
- Methods to polish contigs
Future of assembly

The future of genome sequencing according to MidJourney. Left: yes, but kinda boring. Right: hmmm.
Consensus during sequencing

Start with high-quality double stranded DNA

Ligate SMRTbell adapters and size select

Anneal primers and bind DNA polymerase

Circularized DNA is sequenced in repeated passes

The polymerase reads are trimmed of adapters to yield subreads

Consensus is called from subreads

HiFi data

Stands for ”High Fidelity”

Very low error rates ≈ 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
Human diploid assembly (20/46 chr)

<table>
<thead>
<tr>
<th>Asm</th>
<th>Contig NG50 (Mb)</th>
<th>Scaffold NG50 (Mb)</th>
<th>Hamming error</th>
<th>QV</th>
<th>#Errors Chr X &amp; Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downsampled (35× HiFi / 60× ONT-UL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verkko</td>
<td>12.90</td>
<td></td>
<td>0.13%</td>
<td>52.18</td>
<td>10</td>
</tr>
<tr>
<td>Verkko + trio</td>
<td><strong>80.77</strong></td>
<td><strong>102.55</strong></td>
<td>0.13%</td>
<td>52.40</td>
<td>10</td>
</tr>
<tr>
<td>Verkko + Hi-C</td>
<td>58.24</td>
<td>82.42</td>
<td>0.16%</td>
<td>52.48</td>
<td>10</td>
</tr>
<tr>
<td>LJA</td>
<td>0.39</td>
<td></td>
<td>0.14%</td>
<td>55.75</td>
<td>7</td>
</tr>
<tr>
<td>Hifiasm (unitigs)</td>
<td>0.35</td>
<td></td>
<td>0.23%</td>
<td><strong>61.05</strong></td>
<td>2</td>
</tr>
<tr>
<td>Hifiasm + trio</td>
<td>64.50</td>
<td></td>
<td><strong>0.06%</strong></td>
<td>60.86</td>
<td>26</td>
</tr>
<tr>
<td>Hifiasm + Hi-C</td>
<td>66.34</td>
<td></td>
<td>0.79%</td>
<td>60.57</td>
<td>37</td>
</tr>
<tr>
<td>Full-coverage (&gt;100× HiFi / 85× ONT-UL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verkko</td>
<td>17.35</td>
<td></td>
<td><strong>0.05%</strong></td>
<td>54.91</td>
<td>5</td>
</tr>
<tr>
<td>Verkko + trio</td>
<td><strong>134.00</strong></td>
<td><strong>135.80</strong></td>
<td><strong>0.05%</strong></td>
<td>55.77</td>
<td>5</td>
</tr>
<tr>
<td>Verkko + Hi-C</td>
<td>68.32</td>
<td>85.97</td>
<td><strong>0.05%</strong></td>
<td>55.57</td>
<td>5</td>
</tr>
<tr>
<td>HPRC curated</td>
<td>72.70</td>
<td><strong>146.75</strong></td>
<td>0.13%</td>
<td><strong>61.35</strong></td>
<td>26</td>
</tr>
</tbody>
</table>
Ongoing progress

Errors in Nanopore sequencing data are rapidly diminishing

Q20 chemistry achieved modal accuracy $\approx 99\%$
High fidelity Nanopore incoming?

Nanopore duplex reads could deliver long and precise reads in the future
## Take home messages

- 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
Ongoing 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