References


- Technical note from Illumina:
The classic approach to DNA-Sequence assembly from shotgun fragments consists of three stages:

1. **Overlap**: For each pair of reads it is checked whether they significantly overlap in sequence. This is a local alignment problem since sequencing errors result in indels and mismatches.

2. **Layout**: Reads are placed, i.e. the position of the reads in the assembled sequence are determined. Not all reads may be placed.

3. **Consensus**: For each position of the assembled sequence a (consensus) nucleotide is determined from all reads that align to that position.

Individual programs differ mostly in stages 1) and 2).
Genomanalyse
Mario Stanke

Puzzle Analogy

Assembly compared to solving a Jigsaw-Puzzle

- reads $\hat{\land}$ puzzle pieces
- two overlapping reads are be joined $\hat{\land}$ two puzzle pieces put together
- contig $\hat{\land}$ connected assembled part of the image
- scaffold $\hat{\land}$ several connected assembled parts of which approximate relative position is known
Classic Approach: Overlap-Layout-Consensus

Overlap-Layout-Consensus

- Programs following this approach have been used for most eukaryotic genomes so far (in particular human and mouse).
- Commonly applied programs include CELERA, ARACHNE, Phrap and Atlas.
Limitation of Overlap-Layout-Consensus

Next-Generation-Sequencing

has two major characteristics:

1. the number of reads is much higher than before:
   - before: average coverage 7X (minimum as required by Lander-Waterman)
   - with NGS: average coverage like 50X (Panda)

2. reads are shorter than with Sanger sequencing

1. has the effect that the overlap step of classic assemblers becomes inefficient/infeasible:
   - This step has an inherently quadratic component, each overlap between a pair of reads needs to be considered.

2. means that we do not loose as much information when considering $k$-mers instead of complete reads
New Approach to Assembly

Idea

- too slow to compare pairs of reads, do not try
- consider all \( k \)-mers contained in all reads \((\wedge\text{ break puzzle pieces})\)
- identify overlap of reads through common \( k \)-mers
- join sequence of \( k \)-mers to longer string, if neighboring \( k \)-mers overlap by \( k - 1 \) characters
- use subgraph of a De Bruijn Graph
De Bruijn Graph

- directed graph $G = (V, E)$ representing overlaps between sequences of symbols
- named after the Dutch mathematician Nicolaas Govert de Bruijn

Definition (De Bruijn Graph)

$G = (V, E)$ is a $k$-dimensional De Bruijn graph of $m$ symbols $\Sigma = \{s_1, \ldots, s_m\}$ if

$$V = \Sigma^k$$

and

$$E = \{(v_1, \ldots, v_k), (w_1, \ldots, w_k) \mid v_2 = w_1, v_3 = w_2, \ldots, v_k = w_{k-1}\}.$$
De Bruijn Graph

Vertices are interpreted as sequences: There is an edge between vertices \( v \) and \( w \) iff the suffix of \( v \) of length \( k-1 \) is equal to the prefix of \( w \) of length \( k-1 \).
De Bruijn Graph for Sequence Assembly

**Graph Construction**

- fix \( k \) (e.g. \( k = 21 \))
- break each read \( r \) of length \( \ell \) up into the \( \ell - k + 1 \) overlapping \( k \)-mer substrings of \( r \)
- build a subgraph of a \( k \)-dimensional De Bruijn graph of the 4 Symbols \( \Sigma = \{ A, C, G, T \} \):
  - \( V \) = the set of all \( k \)-mers contained in any read
  - \( E = \{ (v, w) \mid v, w \text{ are } k \text{-mers and the last } k - 1 \text{ chars of } v \text{ are the first } k - 1 \text{ chars of } w \} \)
De Bruijn Graph Example

Example (Toy De Bruijn Graph, $k = 3$)

Read: AGATGATTGC

3-mers:

De Bruijn Graph

AGA -> GAT -> ATG -> TGA
ATT
TTC -> TCG

illumina
Genomanalyse
Mario Stanke

Classic Approach: Overlap-Layout-Consensus

New Approach: De Bruijn Graph

Correspondence Between Paths and Assembled Strings

Reminder: Path

In a directed graph \((V, E)\), a path is a sequence of vertices \((v_1, v_2, \ldots, v_n)\) such that \((v_i, v_{i+1}) \in E\) for \(i = 1, \ldots, n-1\).

A path defines a string

Let \((v_1, v_2, \ldots, v_n)\) be a path in above De Bruijn graph for sequence assembly. Then this path defines the nucleotide sequence

\[ v_1 v_2[k] v_3[k] \cdots v_n[k]. \]

Example (string defined by a path)

The path (AGA, GAT, ATT, TTC, TCG) defines the string AGATTTCG.
Correspondence Between Paths and Assembled Strings

Each read defines a path

Each read $r$ of length $\ell$ naturally defines a path, the sequence of its $k$-mers:

$$(r[1..k], r[2..k + 1], \ldots, r[k - \ell + 1..\ell])$$

The nucleotide sequence defined by this path is $r$ itself.

Example (path defined by a read)

The read AGATGATTCG defines the path (AGA, GAT, ATG, TGA, GAT, ATT, TTC, TCG).
Assembly using De Bruijn Graph

- for understanding concept make simplifying assumptions:
  1. no sequencing errors
  2. strand of reads are known
  3. no paired-end information
  4. complete coverage

- assembly problem can now be posed as problem of finding paths in a graph
- the true target sequence (assembly) corresponds to a path through $G$ that contains every vertex
Another simplifying assumption ...

If all \( k - 1 \)-mers of the target genome \( T \) are unique then the graph is a linear chain of vertices, i.e.

\[
V = \{ \text{all } k\text{-mers of } T \}
\]

\[
E = \{ (v_i, v_{i+1}) \mid v_i \text{ is the } k\text{-mer starting at pos. } i \text{ of } T, 1 \leq i \leq |T| - k + 1 \}.
\]

*Proof:* Exercise

Contigs

- linear chain subgraphs can be identified
- each such chain gives rise to an assembled contig
- assembly unambiguously in that range
Assembly using De Bruijn Graph

Example (ideal case, $k=4$)

target genome: ACTTGACGCGTTACGAATATATCG

reads: ACTTGACGC

GACGCGTTAC

TTACGA

ACGAATAT

ATATCG

De Bruijn graph:

Find the path through all vertices and output the string it defines as assembled genome.
Assembly using De Bruijn Graph

Extensions to solve the real problem

- **strandedness:**
  with every $k$-mer also add the reverse complement to the graph

- **pairedness:**
  use read pairs from paired-end sequencing to join contigs to scaffolds and to resolve ambiguities

- **repeats:**
  use full length reads and read pairs to resolve repeats *(chalk board)*

- **sequencing errors:**
  - remove $k$-mers with frequency below threshold (possibly from sequencing errors)
  - trimming and bubble popping
## Assembly using De Bruijn Graph

**Error correction: trimming**

- sequencing errors at most $k$ positions from a read end result in tips:

### Example (tip)

**Reads:**

- ACTTGACGC
- GACGCGTTTC
- GTTACGA
- ACGAATAT
- ATATCG

**De Bruijn Graph:**

- tips are removed from graph in early stage
Assembly using De Bruijn Graph

Error correction: bubble popping

- sequencing errors or polymorphisms in the middle of reads result in “bubbles”:

Example (bubble)

Reads:

- ACTTGACGC
- GACG\textcolor{red}{T}GTTAC
- CGCGTTAC
- TTACGA
- ACGAATAT
- ATATCG

De Bruijn Graph:

- bubbles are “popped”: replace redundant paths by one, which represents consensus (most coverage)
Assembly using De Bruijn Graph

<table>
<thead>
<tr>
<th>Choice of $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>too small:</td>
</tr>
<tr>
<td>- many ambiguous $k$-mers</td>
</tr>
<tr>
<td>too large:</td>
</tr>
<tr>
<td>- sequencing errors destroy connectivity</td>
</tr>
<tr>
<td>- more memory required</td>
</tr>
<tr>
<td>practical recommendation: parameter scan - try several values for $k$ and pick assembly with long contigs</td>
</tr>
</tbody>
</table>