CS681: Advanced Topics in Computational Biology

Can Alkan
EA224
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs681/
Genome Assembly

Test genome → Random shearing and Size-selection → Sequencing → Contigs/scaffolds → Assemble
De Bruijn Graphs

- **$n$-dimensional directed graph of $m$ symbols**
  - $m^n$ vertices: all possible length-$n$ sequences of $m$ symbols
  - Edges between vertices $v$ and $w$ if $\text{sequence}(w)$ can be generated by shifting $\text{sequence}(v)$ by one character and add one new character
  - $S = \{s_1, s_2, \ldots, s_m\}$
  - $V = S^n = \{(s_1, \ldots, s_1, s_1), (s_1, \ldots, s_1, s_2), \ldots, (s_m, \ldots, s_m, s_m)\}$
  - $E = \{((v_1, v_2, \ldots, v_n), (w_1, w_2, \ldots, w_n)) : v_2 = w_1, v_3 = w_2, \ldots, v_n = w_{n-1}\}$
De Bruijn Graph for DNA Assembly

- \( m = 4 \) (A, C, G, T)
- \( n = k \) (k-mer size)
- \( 4^k \) potential vertices
  - In reality if \( k \) is sufficiently large, upper bound is genome size
  - Twin vertices: vertices with sequences that are reverse-complement of each other
    - AAAAA twin of TTTT
De Bruijn Assemblers

- Currently the most common for NGS: Euler, ALLPATHS-LG, Velvet, ABySS, SOAPdenovo
- Divide reads into k-mers
  - Build graph from k-mers
    - Put an edge if there is k-1 bp prefix-suffix match
  - Error correction
  - Eulerian path
- The first parts (graph construction & correction) is essentially common to all these assemblers, with a few implementation differences (e.g. parallelization in ABySS)
A quick example

```
TAGTCGAGGCTTTTAGATCCGATGAGGCTTTAGAGACAG

AGTCGAG CTTTATA CGATGAG CTTTATA
GTCGAGG TTAGATC ATGAGGC GAGACAG
GAGGCTCT ATCCGAT AGGCTTT GAGACAG
AGTCGAG TAGATCC ATGAGGC TAGAGA
TAGTCGA CTTTATA CGATGAGA TTTAGAG
CGAGGCTC AGATCCG TGAGGCT AGAGACA
TAGTCGA GCTTTATA TCGATG TCTCTAG
TCGACGC GATCCGA GAGGCTT AGAGACA
TAGTCGAT TTAGATC GATGAGG TTTAGAG
GTCGAGG TCTAGAT ATGAGGC TAGAGAC
AGGCTTT ATCCGAT AGGCTTT GAGACAG
AGTCGAG TTAGAT T ATGAGGC AGAGACA
GGCTTTA TCGATG TTTAGAG
CGAGGCT TAGATCC TGAGGCT GAGACAG
AGTCGAG TTTAGATC ATGAGGC TTTAGAG
GAGGCTT GATCCGA GAGGCTT GAGACAG
```

Slide courtesy of Dan Zerbino
A quick example

AGTCGAG CTTTAGA CGATGAG CTTTAGA
GTCGAGG TTAGATC ATGAGGC GAGACAG
GAGGCTC ATCCGAT AGGCTTT GAGACAG
AGTCGAG TAGATCC ATGAGGC TAGAGAA
TAGTCGA CTTTAGA CCGATGA TTAGAGA
CGAGGCT AGATCCG TGAGGCT AGAGACA
TAGTCGA GCTTTAG TCCGATG GCTCTAG
TCGA CGC GATCCGA GAGGCTT AGAGACA
TAGTCGA TTAGATC GATGAGG TTTAGAG
GTCGAGG TCTAGAT ATGAGGC TAGAGAC
AGGCTTT ATCCGAT AGGCTTT GAGACAG
AGTCGAG TTAGATT ATGAGGC AGAGACA
GGCTTTA TCCGATG TTTAGAG
CGAGGCT TAGATCC TGAGGCT GAGACAG
AGTCGAG TTTAGATC ATGAGGC TTAGAGA
GAGGCTT GATCCGA GAGGCTT GAGACAG

Slide courtesy of Dan Zerbino
A quick example

First read: GTCGAGG

Slide courtesy of Dan Zerbino
A quick example

First read: GTCGAGG
Second read: AGTCGAG

Slide courtesy of Dan Zerbino
A quick example

All the others…

Slide courtesy of Dan Zerbino
A quick example

All the others…

Slide courtesy of Dan Zerbino
A quick example

After simplification…

TAGTCGA  CGAG

CGACGC

GATTCGATGAG

GAGGCT

GATT

AGAT

GCTCTAG

TAGA AGAGA AGACAG

GCTTTAG

Slide courtesy of Dan Zerbino
Tips

Slide courtesy of Dan Zerbino
Error removal

Tips removed...

Slide courtesy of Dan Zerbino
Bubbles

Slide courtesy of Dan Zerbino
Error removal

Bubbles removed

Slide courtesy of Dan Zerbino
Error removal

Final simplification…

Slide courtesy of Dan Zerbino
Eulerian path

TAGTCGAG
AGAGACAG
AGATCCGATGAG
GAGGCTTTAGA
TAGTCGAG  GAGGCTTTAGA  AGATCCGATGAG  GAGGCTTTAGA  AGAGACAG

Slide courtesy of Dan Zerbino
Differences: de Bruijn vs Overlap

- **Algebraic difference:**
  - Reads in the OLC methods are atomic
  - Reads in the DB graph are sequential paths through the graph

- **This leads to practical differences:**
  - DB graphs allow for a greater variety of overlaps.
  - Overlaps in the OLC approach require a global alignment, not just a shared $k$-mer
Considerations

- Graph size scales with genome size
  - Increased error rate -> larger graph
- Clipping to short k-mers get rid of sequence errors accumulated at the ends of reads
- k value:
  - Small -> increased connectivity vs. more repeat collapses
  - Large -> increased specificity vs. decreased connectivity
Resolving repeats using long reads or paired-end reads

REPEAT RESOLUTION
Chromosome X

• 548 million Illumina reads were generated from a flow-sorted human X chromosome.
  • Fit in 70GB of RAM.
  • Many contigs: 898,401 contigs
  • Short contigs: 260bp N50 (max 6,956bp)
  • Overall length: 130Mb.

• Moral: there are engineering issues to be resolved but the complexity of the graph needs to be handled accordingly.
  • Reduced representation (Margulies et al.).
  • Combined re-mapping and de novo sequencing (Cheetham et al., Pleasance et al.).
  • Code parallelization (ABySS)
  • Improved indexing (Cortex).
  • Use of intermediate re-mapping
Repeats in a de Bruijn graph
Velvet: RockBand

Use long and short reads together

Slide courtesy of Dan Zerbino
Different approaches to repeat resolution

- Theoretical: spectral graph analysis
  - Equivalent to a Principal Component Analysis
  - Relies on a (massive) matrix diagonalization
  - Comprehensive: all the data is integrated at once
  - Robust: small variations don’t disturb the overall result
  - Never used because of the computational cost.
Different approaches to repeat resolution

- Traditional scaffolding
  - e.g. Arachne, Celera, BAMBUS.
  - Heuristic approach similar to that used in traditional overlap-layout-consensus contigging.
  - Build a big graph of pairwise connections, simplify, extract obvious linear components.
Different approaches to repeat resolution

- In NGS assemblers:
  - **EULER**: for each pair of reads, find all possible paths from one read to the other.
  - **ABySS**: Same as above, but the read-pairs are bundled into node-to-node connections to reduce calculations.
  - **ALLPATHS**: Same as above, but the search is limited to localized clouds around pre-computed scaffolds.

*Slide courtesy of Dan Zerbino*
Different approaches to repeat resolution

- Using the differences between insert length
  - The Shorty algorithm uses the variance between read pairs anchored on a common contig on $k$-mer.

Collapsed repeat in contig1?
PRACTICAL CONSIDERATIONS
Colors pace

- Di-base encoding has a 4 letter alphabet, but very different behavior to sequence space
  - Different rules for complementarity
- Direct conversion to sequence-space is simple but erroneous
  - One error messes up all the remaining basepairs
- Conversion must therefore be done at the very end of the process, when the reads are aligned
  - You can then use the transition rules to detect errors
Different error models

- When using different technologies, you have to take into account different technologies
  - Easy for OLC assembly
  - Much more tricky for de Bruijn assembly, since k-mers are not assigned to reads.
  - Different assemblers have different settings
Pre-filtering the reads

- Some assemblers have built-in filtering of the reads (e.g. Euler) but not a generality.
  - Low phred quality
  - Reads with N characters
- Efficient filtering of low quality bases can cut down on the computational cost (memory & time)
- Some assemblers require reads of identical lengths.

*Slide courtesy of Dan Zerbino*