Assembly and Alignment

Workshop on Comparative Genomics

İnanç Birol, Shaun Jackman

Smithsonian Institution – 5 October 2011
Assembly and Alignment

Workshop on Comparative Genomics

İnanç Birol, Shaun Jackman

Smithsonian Institution – 5 October 2011
Moore’s Law

The graph illustrates the exponential growth of transistor count and computational power from 1900 to 2000. The x-axis represents the year, while the y-axis shows the transistor count and calculations per second per $1,000. The graph highlights the transition from electromechanical to integrated circuits, showing a significant increase in computational capabilities.
Growth of Knowledge

# Citations / Article

Periods

Periods

# Citations

Millions
Next Generation Sequencing

- Illumina sequencing throughput at GSC
- Cost of sequencing human genome

![Graph showing Illumina sequencing throughput from May 2007 to Aug 2010 with two lines: PF_3month ave and Raw_3month ave.]

![Graph showing the cost of sequencing a human genome from 2003 to 2010, with a steep decline represented by a red line without a specific unit of currency indicated.]
Old paradigm:
- long and non-uniform reads (800bp - 1000bp)
Old paradigm:

- long and non-uniform reads (800bp - 1000bp)
- overlap; overlay; consensus
De Novo Assembly Problem

- New paradigm:
  - short and uniform reads (50bp - 150bp)
De Novo Assembly Problem

- New paradigm:
  - short and uniform reads (50bp - 150bp)
  - overlap; overlay; consensus
New paradigm:
- short and uniform reads (50bp - 150bp)
- de Bruijn graphs
De Novo Assembly Problem

- New paradigm:
  - long range information through read pairs
  - graph theoretic approaches
Assembly By Short Sequencing
• **SE Assembly**: k-mer extension on a de Bruijn graph

• **PE Assembly**: search for unambiguous contig merging along paths

• **Scaffolding**: search for unambiguous linkage across distant contigs
Genome builders face the competition

Three independent projects seek to contrast approaches in preparation for routine analysis of genetic data.

Erika Check Hayden

Sequencing DNA on an industrial scale is no longer difficult: the challenge is in assembling a full genome from the multitude of short, overlapping snippets that second-generation sequencing machines can generate. At a meeting last week at the University of California, Santa Cruz, three winners emerged: ALLPATHS-LG, developed by the Broad Institute in Cambridge, Massachusetts; ABYSS, developed at the British Columbia Cancer Agency's Genome Sciences Centre in Vancouver, Canada; and SOAPdenovo, developed by the Beijing Genomics Institute. But, Korf notes, "it's not just the software, it's how people are running it" that determines the quality of each assembly.
A **partial** and **unambiguous** read-to-read alignment extends the length of sequence information

- First stage of an assembly algorithm is to find such alignments
- Assembly algorithms differ in the way they find and use these alignments
Greedy Assembly

• Find two reads with the largest overlap
• Merge them

Repeat until no more

Pro: fast
Con: prone to misassembly
• Assumes largest overlaps are unambiguous
Overlap Overlay Consensus

- **Overlap**
  Find all pairs of sequences that overlap

- **Overlay (a.k.a. Layout)**
  Remove redundant and weak overlaps

- **Consensus**
  Merge pairs of sequences that overlap unambiguously
  Build a consensus sequence from all reads overlaid in a region
Find Overlapping Reads

• Naïve algorithm: make all binary comparisons
  Untenable when too many reads
  – $O(n^2)$
  – RAM
  – CPU

• Ferragina-Manzini index
  – Apply Burrows-Wheeler transform
  – Small memory footprint

Build an overlap graph
Forget About Overlapping Reads!

- Shred reads to a uniform length $k$
- Build a special overlap graph: de Bruijn Graph

$$\text{read1: 20-mer}$$

```
TCGATCGATTTTCGGCCTAA
TCGATCGATTTTCG
CGATCGATTTTCGG
GATCGATTTTCGGC
ATCGATTTTCGGCC
TCGATTTTCGGCT
CGATTTTCGGCCTA
GATTTTCGGCCTAA
```

seven 14-mers

$$\text{read2 : 20-mer}$$

```
ATTTTCGGCCCTAAATTTAGG
ATTTTCGGCCCTAAT
TTTTGGCCTAAATA
TTTCGGCCTAAATAT
TTCGCCTAAATT
TCGGCCTAAATTTA
CGGCTAAATTAG
GGCCTAAATTAG
```

seven 14-mers

```
...GCAATCGATCGATTTTCGGCCTAAATTTAGGCCGATAATCGACGATC...
```
De Bruijn Graph

• Load $k$-mers in memory
  – 2x4 possible extension of every $k$-mer
• Check if there is a “next” $k$-mer
  – $O(n)$ algorithm

...GACATTGC... seq1
...GACATTAT... seq2

\[ k = 5 \]
Memory Concerns

• Human genome has over 2 billion unique $k$-mers
• If we represent every $k$-mer using, say 50 bytes, we require over 100 GB RAM just to represent $k$-mers

Solution #1: Clustering reads
   Curtain (w/ Velvet)
   Phusion (w/ Phrap)

Solution #2: Distributed computing
   ABySS
   SOAPdenovo
   ALLPATH-LG
Partitioning Read Space

Distribute sub-reads and reverse-complements over nodes

\begin{itemize}
  \item \textit{n}-mer read
  \item \textit{k}-mers
  \item Read length \( n = 36 \)
  \item Hash key length \( k = 26 \)
\end{itemize}

\begin{align*}
\text{TTG\textit{CATCGATCGATTTATCGGCCCTAACTATAATTACC}} & \quad \text{node} \quad 94 \\
\text{TTG\textit{CATCGATCGATTTATCGGCCCTAAT}} & \quad 149 \\
\text{GCATCGATCGATTTATCGGCCCTAAT} & \quad 40 \\
\text{CATCGATCGATTTATCGGCCCTAATC} & \quad 19 \\
\text{ATCGATCGATTTATCGGCCCTAATCT} & \quad 27 \\
\text{TCGATCGATTTATCGGCCCTAATCTA} & \quad 0 \\
\text{CGATCGATTTATCGGCCCTAATCTA} & \quad 87 \\
\text{GATCGATTTATCGGCCCTAATCTATT} & \quad 145 \\
\text{ATCGATTTATCGGCCCTAATCTATT} & \quad 128 \\
\text{TCGATTTATCGGCCCTAATCTATTAC} & \quad 84 \\
\text{CGATTTATCGGCCCTAATCTATTACC} & \quad 106
\end{align*}

\begin{align*}
\text{GCATCGATCGATTTATCGGCCCTAAT} & \quad 100100110100011011001111100110101001011011000011 \\
\text{ATTAGGGCCGATAAAATCGATCGATGC} & \quad 00111001010100101100011000000110110011011000111001 \\
\text{mod} \ 160 & \quad 10101111110010100000000101111111100000101000111111110 \\
\text{\text{XOR}} & \quad 40
\end{align*}
Graph Generation

• A given $k$-mer can have up to 8 extensions
• Each node announces the list of $k$-mers that it has to the nodes that hold their possible extensions
• Each node records if there are any extensions of the $k$-mers that it stores
• This forms adjacency information for $k$-mers over a distributed de Bruijn graph
Trimming

- Data would have experimental noise
- de Bruijn graph would have false branches
- Some read errors are filtered by removing such branches
- Trimming prevents the later assembly step to come to a premature end because of read errors
**Bubble Popping**

- Repeat read errors and single nucleotide allelic differences would cause “bubbles” of length $2^k-1$
- Bubbles are popped by removing either of those branches
- Complex bubbles can form when multiple bubbles intersect
  - Bubble popping step either reduces the bubble orders by one
  - Or creates dead branches
- Popped bubbles are recorded in a log file to study potential allelic differences
• Remaining de Bruijn graph is analyzed for contig extension ambiguities
• If there is a multiplicity in the inbound or outbound contig extensions, then contig growth is terminated
• SET assembly step then concatenates the remaining connected nodes in the di-graph, creating independent contigs that overlap by no more than \( k-1 \) bases
Assembly - PET

- After SET assembly, reads are aligned to contigs
- Using reads that hit the same contig, empirical fragment size distribution(s) is (are) calculated
- Using reads that hit multiple contigs, inter-contig distances are inferred with a maximum likelihood estimator
- Contigs with coherent and unambiguous distances are joined
• SET assembly result as a graph
  – Nodes: contigs
  – Edges: overlaps (k-1 bp)
• SET assembly result as a graph
  – Nodes: overlaps (k-1 bp)
  – Edges: contigs
Assembly As a Hairball
ABySS-Explorer: Visualizing Genome Sequence Assemblies

Cydny B. Nielsen, Shaun D. Jackman, Inanç Birol, and Steven J.M. Jones

Contig Length

1 oscillation = 100 bp
Paired End Tag Information

inbound partners (light orange)  selected contig (orange)  outbound partners (dark orange)
Paired End Contigs

Blue gradient: SET contig path in PET assembly
Orange: selected SET contig
ABySS-Explorer GUI
Statistics Display
Nxx plot and N50

- The N50 is the weighted median of contig sizes.
- The N50 summarizes a single point on the Nxx plot.
- Better assemblies are further to the right.
**k-mer Coverage Histogram**

- Counts the number of occurrences of each k-mer
- Useful for
  - estimating the genome size
  - measuring mean coverage
  - library quality control
ABySS
Assembly By Short Sequences - a de novo, parallel, paired-end sequence assembler

Current release
ABySS 1.3.0
Released Sep 09, 2011

Mate-pair data can be used to scaffold contigs. Specify your mate-pair libraries using the 'mp' parameter of abyss-pe.

More about this release...

Get ABySS for all platforms (493 KB)
Source

Get ABySS for Linux (1.2 MB)
Debian package (amd64)

Project Description
ABySS is a de novo, parallel, paired-end sequence assembler that is designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size. The parallel version is implemented using MPI and is capable of assembling larger genomes.

To assemble transcriptome data, see Trans-ABySS.

Publications
Assemble the input files

AbYSS

collection.hist  ecoli-1.fa  ecoli-bubbles.fa

Find overlaps of (m,k) bases

AdjList

ecoli-1.adj  ecoli-1.path

Identify and pop simple bubbles

PopBubbles

ecoli-3.adj  ecoli-1.path

Merge paths of contigs to create larger contigs

awk

ecoli-indel.fa

Align reads to contigs

abyss-map

abyss-fixmate

Estimate distances between contigs

DistanceEst

Merge distance estimates

abyss-joindist
Assembly Operations

• SET contig building: de Bruijn
  – $k$-mer overlap information
• SET error removal: adjacency
• PET contig merging: adjacency & linkage
  – PET alignments
• PET/MPET scaffolding: adjacency & linkage
  – PET/MPET alignments
• Gap closure and contig extensions: read overlap
  – PET alignments
1. Erode low-coverage tips

- e, --erode=COVERAGE
  
erode bases at the ends of blunt contigs with coverage less than this threshold

- E, --erode-strand=COVERAGE
  
erode bases at the ends of blunt contigs with coverage less than this threshold on either strand
2. Trim tips

- `t, --trim-length=TRIM_LENGTH`
  - maximum length of dangling edges to trim
3. Remove low coverage contigs

-\texttt{-c, --coverage=COVERAGE}

remove contigs with mean k-mer coverage less than this threshold
4. Pop bubbles

- `b`, `--bubbles=N`
  
  pop bubbles shorter than N bp (default: 3*k)

- `b0`, `--no-bubbles`
  
  do not pop bubbles
Scaffold Graph Operations

1. Resolve forks
2. Trim tips
3. Remove repeats
4. Remove transitive edges
Scaffold Graph Operations

5. Trim tips
Scaffold Graph Operations

6. Pop bubbles
Scaffold Graph Operations

7. Remove weak edges
Running ABySS

- Assemble the paired-end reads in the file reads.fa
  
  > abyss-pe name=ecoli k=32 n=10 in=reads.fa

- Assemble the paired-end reads in the files reads_1.fa and reads_2.fa:
  
  > abyss-pe name=ecoli k=32 n=10 in='reads_1.fa reads_2.fa'
Running ABySS in Parallel

- Run ABySS using eight threads
  > abyss-pe np=8 name=ecoli k=32 n=10 in='reads_1.fa reads_2.fa'

- ABySS uses MPI, the Message Passing Interface. OpenMPI is an open-source implementation of MPI
Running Parallel Jobs on a Cluster

- Run ABBySS on a cluster using 8 threads
  
  ```shell
  > qsub -pe openmpi 8 -N ecoli
  abyss-pe np=8 name=ecoli k=32 n=10
  in='reads_1.fa reads_2.fa'
  ```

- `abyss-pe` uses the environment variables `JOB_NAME` and `NSLOTS` passed to it by SGE as the default values for `name` and `np`
Assemble every 8\(^{th}\) \(k\) from 32 to 96

\[
qsub \ -pe \ openmpi \ 8 \ -N \ ecoli \ -t \ 32-96:8 \ abyss-pe \ k=32 \ n=10 \ in=\textquote{reads}_1.fa \ reads_2.fa
\]

- abyss-pe uses the environment variable \textit{SGE\_TASK\_ID} passed to it by SGE as the default value for \(k\)
Assembling Multiple Libraries

```bash
> abyss-pe name=ecoli
  k=32  n=10
  lib='pe200  pe500'
  pe200='pe200_1.fa  pe200_2.fa'
  pe500='pe500_1.fa  pe500_2.fa'
```
Assembling a Mix of PET and SET

> abyss-pe name=ecoli
  k=32  n=10
  lib='pe200  pe500'
  pe200='pe200_1.fa  pe200_2.fa'
  pe500='pe500_1.fa  pe500_2.fa'
  se='long.fa'
Parameters of ABYSS

- **name**: name of the assembly
- **lib**: name of the libraries (one or more)
- **se**: paths of the single-end read files
- **${lib}**: paths of the read files for that library

**Example**

```bash
> abyss-pe name=ecoli k=32 n=10
lib='pe200 pe500'
pe200='pe200_1.fa pe200_2.fa'
pe500='pe500_1.fa pe500_2.fa'
se='long.fa'
```
Parameters of ABySS (SET)

- **k**: the size of a k-mer
- **q**: quality trimming removes low-quality bases from the ends of reads
- **e** and **c**: coverage-threshold parameters
  - **e**: erosion removes bases from the ends of contigs
  - **c**: coverage threshold removes entire contigs
- **p**: the minimum identity for bubble popping
Parameters of ABYSS (PET)

- **s**: the minimum size of a seed contig
- **n**: the number of pairs required to join two contigs

**Example**

```plaintext
> abyss-pe name=ecoli
  k=64 q=3 p=0.9 s=100 n=10
lib='pe200 pe500'
pe200='pe200_1.fa pe200_2.fa'
pe500='pe500_1.fa pe500_2.fa'
se='long.fa'
```
Optimizing $k$

- Assemble every 8\textsuperscript{th} $k$ from 32 to 96
  Nine assemblies: 32 40 48 56 64 72 80 88 96
- Find the peak
- Assemble every 2\textsuperscript{nd} $k$ around the peak
  For example, if the peak were at $k=64$...
  Eight assemblies: 56 58 60 62 66 68 70 72
- SGE:
  
  > qsub -t 32-96:8 qsub-abyss.sh
  > qsub -t 56-72:2 qsub-abyss.sh
Output Files of ABySS

- **${name}-contigs.fa**
  The final contigs in FASTA format

- **${name}-bubbles.fa**
  The equal-length variant sequences (FASTA)

- **${name}-indel.fa**
  The different-length variant sequences (FASTA)

- **${name}-contigs.dot**
  The contig overlap graph in Graphviz format
Intermediate Output Files of ABySS

- `.adj`: contig overlap graph in ABySS adj format
- `.dist`: estimates of the distance between contigs in ABySS dist format
- `.path`: lists of contigs to be merged
- `.hist`: fragment-size histogram of a library
- `coverage.hist`: k-mer coverage histogram
Case Study

Mountain Pine Beetle Genome Assembly
Mountain Pine Beetle Genome

Assembly statistics

<table>
<thead>
<tr>
<th></th>
<th>contigs</th>
<th>scaffolds</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>1,128,463</td>
<td>1,103,221</td>
</tr>
<tr>
<td>n:500bp</td>
<td>33,591</td>
<td>11,657</td>
</tr>
<tr>
<td>n:N50</td>
<td>4,324</td>
<td>82</td>
</tr>
<tr>
<td>N50 (bp)</td>
<td>11,220</td>
<td>541,443</td>
</tr>
<tr>
<td>Max (bp)</td>
<td>276,135</td>
<td>3,583,207</td>
</tr>
<tr>
<td>Reconstruction (Gb)</td>
<td>201.9</td>
<td>200.4</td>
</tr>
</tbody>
</table>
• ABySS v1.2.7
  – PET/MPET information disambiguates short contig extensions

<table>
<thead>
<tr>
<th></th>
<th>in 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15822</td>
<td>7354</td>
<td>1882</td>
<td>530</td>
<td>109</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>7354</td>
<td>9814</td>
<td>1817</td>
<td>456</td>
<td>72</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1882</td>
<td>1817</td>
<td>1074</td>
<td>238</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>530</td>
<td>456</td>
<td>238</td>
<td>126</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>109</td>
<td>72</td>
<td>31</td>
<td>13</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>6+</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

* For contigs ≥ 2 kb
• Contig 4 is (eventually) followed by Contig 7
Biotin Read-Through

circularized insert
Dr. Strangelove
Or: How I Learned to Stop Worrying and Love The Biotin Read-Through
Triage of MPET Reads

Challenge:

Information:

- Distances from contig ends
- Base mismatches on read ends
- Inferred contig orientations
# Triage of MPET Reads

<table>
<thead>
<tr>
<th>Read 1</th>
<th>Read 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="MPET-like diagram" /></td>
<td><img src="image2" alt="MPET-like diagram" /></td>
<td>MPET-like</td>
</tr>
<tr>
<td><img src="image3" alt="PET-like diagram" /></td>
<td><img src="image4" alt="PET-like diagram" /></td>
<td>PET-like</td>
</tr>
<tr>
<td><img src="image5" alt="MPET-like diagram" /></td>
<td><img src="image6" alt="PET-like diagram" /></td>
<td>MPET-like</td>
</tr>
<tr>
<td><img src="image7" alt="PET-like diagram" /></td>
<td><img src="image8" alt="PET-like diagram" /></td>
<td>PET-like</td>
</tr>
<tr>
<td><img src="image9" alt="MPET-like diagram" /></td>
<td><img src="image10" alt="PET-like diagram" /></td>
<td>MPET-like</td>
</tr>
<tr>
<td><img src="image11" alt="PET-like diagram" /></td>
<td><img src="image12" alt="PET-like diagram" /></td>
<td>PET-like</td>
</tr>
</tbody>
</table>

*Note: Images are placeholders and should be replaced with actual diagrams.*
Scaffolding
• Scrubbing “homozygous” variations

Indel (2,935)

SNPs (19,715)

www.bcgsc.ca
• Local directional assembly
  – scaffold gap filling
  
  (10,499 of 63,986)

  – extension

  (20,213 of 53,487)
## Quality Assessment

### Alignment of 81,047,980 reads

<table>
<thead>
<tr>
<th></th>
<th>Before Anchor</th>
<th>After Anchor</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapped</td>
<td>65,624,456</td>
<td>66,949,341</td>
<td>+ 1,324,885</td>
</tr>
<tr>
<td></td>
<td>(80.97%)</td>
<td>(82.60%)</td>
<td></td>
</tr>
<tr>
<td>Paired</td>
<td>43,207,118</td>
<td>44,732,320</td>
<td>+ 1,525,202</td>
</tr>
<tr>
<td></td>
<td>(53.31%)</td>
<td>(55.19%)</td>
<td></td>
</tr>
<tr>
<td>Single-end</td>
<td>9,536,178</td>
<td>8,846,977</td>
<td>-689,201</td>
</tr>
<tr>
<td></td>
<td>(11.77%)</td>
<td>(10.92%)</td>
<td></td>
</tr>
</tbody>
</table>

### Gene alignments

<table>
<thead>
<tr>
<th></th>
<th>2,180 ESTs</th>
<th>248 Conserved Genes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complete</td>
<td>Partial</td>
</tr>
<tr>
<td>Contigs</td>
<td>968</td>
<td>1169</td>
</tr>
<tr>
<td>Scaffolds</td>
<td>1,481</td>
<td>619</td>
</tr>
</tbody>
</table>
Final Hairball

• ABYSS v1.2.7
  – Read pairs and inferred distances allow for scaffolding

<table>
<thead>
<tr>
<th></th>
<th>contigs</th>
<th>scaffolds</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>1,128,463</td>
<td>1,103,221</td>
</tr>
<tr>
<td>n:500bp</td>
<td>33,591</td>
<td>11,657</td>
</tr>
<tr>
<td>n:N50</td>
<td>4,324</td>
<td>82</td>
</tr>
<tr>
<td>N50 (bp)</td>
<td>11,220</td>
<td>541,443</td>
</tr>
<tr>
<td>Max (bp)</td>
<td>276,135</td>
<td>3,583,207</td>
</tr>
<tr>
<td>Reconstruction (Gb)</td>
<td>201.9</td>
<td>200.4</td>
</tr>
<tr>
<td>Date</td>
<td>ABySS Version</td>
<td>Data</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>August 2009</td>
<td>1.0.11</td>
<td>3x GAiix</td>
</tr>
<tr>
<td>November 2009</td>
<td>1.0.15</td>
<td>+2x GAiix</td>
</tr>
<tr>
<td>February 2010</td>
<td>1.1.1</td>
<td>+4x GAiix</td>
</tr>
<tr>
<td>July 2010</td>
<td>1.2.0</td>
<td>+2x GAiix</td>
</tr>
<tr>
<td>November 2010</td>
<td>1.2.4</td>
<td>+1x GAiix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1x GAiix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MPET)</td>
</tr>
<tr>
<td>May 2011</td>
<td>1.2.7</td>
<td>--</td>
</tr>
<tr>
<td>July 2011</td>
<td>1.2.7</td>
<td>+ 1x HiSeq</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+1x HiSeq</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MPET)</td>
</tr>
<tr>
<td>August 2011</td>
<td>1.2.7</td>
<td>--</td>
</tr>
</tbody>
</table>
Future Work

• Clean up the chaff
  – Place short contigs on Anchored scaffolds
  – Annotate repeat elements

“Finish” assembly
  – Amplicon sequencing
  – Long reads (eg. PacBio)
  – Long-range spatial information (eg. transcriptome)
Transcriptome Assembly
Transcriptome Sequencing

- RNA-seq protocol
- Brings information on how a genome “acts”
  - Expression levels
    - Allelic expression
  - Present isoforms
  - Gene fusions
  - Other transcriptional events
  - Post-transcriptional RNA editing
Transcriptome assembly is different from genome assembly:

- varying coverage levels ⇒ varying expression levels
- split assembly paths ⇒ isoforms/splice variants
- small contig sizes ⇒ small product sizes
What Overlap to Choose?

![Graph showing the relationship between N (thousands) and N(L≥100 bp) (thousands) with respect to k.](chart.png)
What Overlap to Choose?

- Selection of parameter $k$ depends on read coverage depth
- Expression levels vary over 5 orders of magnitude
Selection of $k$

specific $k$  sensitive $k$
Assembly Merging

![Diagram of assembly merging](image)
We capture a wide range of expression levels

- **Gray**: all transcripts with a read alignment
- **Blue**: at least 80% of a transcript in a single contig
- **Red**: at least 80% of a transcript is reconstructed
Trans-ABySS

A versatile tool for

- Transcript reconstruction
- Gene identification
- InDel and SNV discovery
- Chimeric transcript discovery
  - Gene fusions
  - Trans-splicing
- Expression analysis
Transcriptome Assembly

*De novo* assembly based on ABysS

Reference-based assembly based on TopHat alignments

[Trapnell et al., 2010; Guttman et al., 2010; Trapnell et al., 2009]
+ chimeric transcripts
Detecting Fusions

• Conventionally detected through identifying translocations in genomes
• Assembled transcriptome contigs span multiple genes
• Break points (usually) correspond to exon boundaries
• Break points are supported by
  – Spanning reads
  – Read pairs linking regions
Detecting Partial Tandem Duplications

- One or more exons get repeated in their entirety
- Usually coexist with the wild-type
- PTD events are manifested in a particular contig type
  - A short contig with 50/50 split alignment
- Break points are supported by
  - Spanning reads
  - Read pairs in opposite orientation

Lucas Swanson, Readman Chiu and Gordon Robertson
Detecting Internal Tandem Duplications

- Tandem duplications internal to exons
- Contig alignments result in
  - Query gaps
  - Contiguous target blocks
- Read support on break point(s)
- Aberrant read pair distances

Lucas Swanson, Readman Chiu and Gordon Robertson
Performance

• Compared to mapping-based analysis tools
  Trans-ABySS constructs
  – as many transcripts
  – with better sensitivity and specificity

[Trapnell et al., 2010; Guttman et al., 2010; Trapnell et al., 2009]
Sequence Alignment
### Short-Read Sequence Alignment

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFAST</td>
<td>Explicit time and accuracy tradeoff with a prior accuracy estimation, supported by indexing the reference sequences. Optimally compresses indexes. Can handle billions of short reads. Can handle insertions, deletions, SNPs, and color errors (can map ABI SOLID color space reads). Performs a full Smith Waterman alignment.</td>
</tr>
<tr>
<td>BLASTN</td>
<td>BLAST's nucleotide alignment program, slow and not accurate for short reads, and uses a sequence database (EST, sanger sequence) rather than a reference genome.</td>
</tr>
<tr>
<td>BLAT</td>
<td>Made by Jim Kent. Can handle one mismatch in initial alignment step.</td>
</tr>
<tr>
<td>Bowtie</td>
<td>Uses a Burrows-Wheeler transform to create a permanent, reusable index of the genome; 1.3 GB memory footprint for human genome. Aligns more than 25 million Illumina reads in 1 CPU hour. Supports Maq-like and SOAP-like alignment policies (can be run from inside Geneious Server).</td>
</tr>
<tr>
<td>BWA</td>
<td>Uses a Burrows-Wheeler transform to create an index of the genome. It's a bit slower than bowtie but allows indels in alignment (can be run from inside Geneious Server).</td>
</tr>
<tr>
<td>CASHX</td>
<td>Quantify and manage large quantities of short-read sequence data. CASHX pipeline contains a set of tools that can be used together or as independent modules on their own. This algorithm is very accurate for perfect hits to a reference genome.</td>
</tr>
<tr>
<td>CUDA-EC</td>
<td>Short-read alignment error correction using GPUs.</td>
</tr>
<tr>
<td>drFAST</td>
<td>Read mapping alignment software that implements cache obliviousness to minimize main/cache memory transfers like mrFAST and mrsFAST, however designed for the SOLID sequencing platform (color space reads). It also returns all possible map locations for improved structural variation discovery.</td>
</tr>
<tr>
<td>ELAND</td>
<td>Implemented by Illumina. Includes ungapped alignment with a finite read length.</td>
</tr>
<tr>
<td>GNUMAP</td>
<td>Accurately performs gapped alignment of sequence data obtained from next-generation sequencing machines (specifically that of Solexa/Illumina) back to a genome of any size. Includes adaptor trimming, SNP calling and Bisulfite sequence analysis.</td>
</tr>
<tr>
<td>GEM</td>
<td>High-quality alignment engine (exhaustive mapping, that is 100% of sensitivity, for any number of substitutions; 1 non-exhaustive indel). Several standalone applications (mapper, split mapper, mappability, and other) provided.</td>
</tr>
<tr>
<td>GMAP and GSnap</td>
<td>Robust, fast, short-read alignment. GMAP: longer reads, with multiple indels and splices (see entry above under Genomics analysis); GSnap: shorter reads, with a single indel or up to two splices per read. Useful for digital gene expression, SNP and indel genotyping. Developed by Thomas Wu at Genentech. Used by the National Center for Genome Resources (NCGR) in Alpheus.</td>
</tr>
<tr>
<td>Geneious Assumer</td>
<td>Fast, accurate overlap assembler with the ability to handle any combination of sequencing technology, read length, any pairing orientations, with any spacer size for the pairing, with or without a reference genome.</td>
</tr>
<tr>
<td>LAST</td>
<td></td>
</tr>
<tr>
<td>MAQ</td>
<td>Ungapped alignment that takes into account quality scores for each base (can be run from inside Geneious Server).</td>
</tr>
<tr>
<td>mrFAST and mrsFAST</td>
<td>Gapped (mrFAST) and ungapped (mrsFAST) alignment software that implements cache obliviousness to minimize main/cache memory transfers. They are designed for the Illumina sequencing platform and they can return all possible map locations for improved structural variation discovery.</td>
</tr>
</tbody>
</table>
Sequence alignment

- Global
- Local
- Glocal
Global alignment

- Base-by-base alignment of one sequence to another allowing for both mismatches and gaps
- Example:
  \[
  \text{AGAGTGCTGCGCC}
  \]
  \[
  \text{AGATGTACTGCGCC}
  \]
- Alignment:
  \[
  \text{AGA-\textcolor{red}{TGCTGCGCC}}
  \]
  \[
  \text{AGATGTA\textcolor{blue}{ACTGC-GCC}}
  \]
- 12 matches of 15 bp = 80% identity
Local Alignment

- Given two sequences, find a matching substring from each of those two sequences
- Example:
  
  ```
  AGATGTGCTGCCGCC
  TTTGTACTGAAA
  AGATGTGCTGCCGCC
  |   |   |   |
  TTTGTACTGAAA
  ```
  
  - 6 matches of 7 bp = 86% identity
Given a query sequence and a reference sequence, identify a substring of the reference sequence that matches the entirety of the query sequence.

Example:
Reference: AGATGTGCTGCGCGCCACGT
Query: TTTGTACTGAAA
ACGTAAGATGTGCTGCGCGCCACGT

6 matches of 12 bp = 50% identity
Criteria for Choosing an Aligner

- Global, local or glocal alignment
- Aligning short sequences to long sequences
- Aligning long sequences to long sequences
- Handling small gaps (insertions and deletions)
- Handling large gaps (introns)
- Handling split alignments (chimera)
- Speed and ease of use
Popular Alignment Software

Short reads
- BWA
- GSNAP
- Bowtie
  - TopHat
- SOAP

Long sequence
- BWA-SW
- GMAP
- BLAT
- BLAST
- Exonerate
- MUMmer
Seed and Extend

- For large sequences, an exhaustive alignment is very slow
- Many aligners start by finding perfect or near perfect matches to seeds
- The seeding strategy has a large effect on the sensitivity of the aligner
  - e.g. BLAT requires two perfect nearby 11-mer matches
Memory Use

Hashing

- Load a representation of all the reads and/or the reference into memory
  - GSNAP
  - SOAP
  - mr/mrsFAST
  - KAligner

Burrows-Wheeler Transformation
(Ferragina-Manzini, indexing)

- Compress reads and/or the reference before loading
  - BWA
  - Bowtie
  - Abyss-map
Hashing

- TGCATCTCGATTACGGCCCT
- ATCAACATCGTCTTTACCT
- GCTCATTATCCATACATCTA
- CGATATGGCCAATCTATTAC

Hashing Function

01 02 03 04
## BW Transform

<table>
<thead>
<tr>
<th>Rotate</th>
<th>Sort</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGCACT$</td>
<td>$TGCACT</td>
<td>T</td>
</tr>
<tr>
<td>GCACT$T</td>
<td>ACT$TGC</td>
<td>C</td>
</tr>
<tr>
<td>CACT$TG</td>
<td>CACT$TG</td>
<td>G</td>
</tr>
<tr>
<td>ACT$TGC</td>
<td>CT$TGC</td>
<td>A</td>
</tr>
<tr>
<td>CT$TGC</td>
<td>GCACT$T</td>
<td>T</td>
</tr>
<tr>
<td>T$TGC</td>
<td>T$TGCAC</td>
<td>C</td>
</tr>
<tr>
<td>$TGCACT</td>
<td>TGCACT$</td>
<td>$</td>
</tr>
</tbody>
</table>
Inverse BW Transform

Index

T  C  G  A  T  C  $  prepend

round 1
$ T  A  C  C  T  G  
A  C  T  C  A  C  
C  A  C  T  T  $  
C  T  $  T  T  G  
G  C  T  G  C  A  
T  $  T  G  
T  G  

round 2
$ T  G  
A  C  T  
C  A  C  
C  T  $  
C  T  $  T  
G  C  A  C  
T  $  T  G  
T  G  C  A  

round 3
$ T  G  C  
A  C  T  $  
C  A  C  T  
C  T  $  T  
G  C  A  C  
T  $  T  G  
T  G  C  A  

round 4
$ T  G  C  A  
A  C  T  $  T  
C  A  C  T  $  
C  T  $  T  G  
G  C  A  C  T  
T  $  T  G  C  
T  G  C  A  

round 5
$ T  G  C  A  C  
A  C  T  $  T  G  
C  A  C  T  $  T  
C  T  $  T  G  C  
G  C  A  C  T  $  
T  $  T  G  C  A  
T  G  C  A  C  

round 6
$ T  G  C  A  C  T  
A  C  T  $  T  G  C  
C  A  C  T  $  T  G  
C  T  $  T  G  C  A  
G  C  A  C  T  $  T  
T  $  T  G  C  A  C  
T  G  C  A  C  T  $
**Summary**

**De Novo Assembly Problem**
- Old paradigm:
  - long and non-uniform reads (800bp - 1000bp)

**De Bruijn Graph**
- Load k-mers in memory
  - 2x4 possible extension of every k-mer
- Check if there is a “next” k-mer
  - O(n) algorithm
  - \[ \text{GACATGGC} \text{ seq1} \]
  - \[ \text{GACATTAT} \text{ seq2} \]

\[ \cdots \text{GACAT} \text{ ACATT} \text{ CATTA} \text{ ATTAT} \cdots \]

\[ k = 5 \]

**Adjacency Graph**
- SET assembly result as a graph
  - Nodes: overlaps (k-1 bp)
  - Edges: contigs

**ABySS-Explorer GUI**
Summary

Assembly As a Hairball

- ABYSS v1.2.7
  - PET/MPET information disambiguates short contig extensions

Node connectivity

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6+</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15823</td>
<td>734</td>
<td>1882</td>
<td>530</td>
<td>109</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>7356</td>
<td>9814</td>
<td>1817</td>
<td>456</td>
<td>72</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1882</td>
<td>1817</td>
<td>1074</td>
<td>238</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>530</td>
<td>456</td>
<td>238</td>
<td>126</td>
<td>33</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>109</td>
<td>72</td>
<td>31</td>
<td>13</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>6+</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

* For contigs ≥ 2 kb

Biotin Read-Through

Scaffold Graph Operations

1. Resolve forks

Anchor

- Local directional assembly
  - scaffold gap filling
    - (10,499 of 63,986)
  - extension
    - (20,213 of 53,487)
Summary

Trans-ABySS

A versatile tool for
- Transcript reconstruction
- Gene identification
- InDel and SNV discovery
- Chimeric transcript discovery
  - Gene fusions
  - Trans-splicing
- Expression analysis

Detecting Fusions

- Conventionally detected through identifying translocations in genomes
- Assembled transcriptome contigs span multiple genes
- Break points (usually) correspond to exon boundaries
- Break points are supported by
  - Spanning reads
  - Read pairs linking regions

BW Transform

<table>
<thead>
<tr>
<th>Rotate</th>
<th>Sort</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGCAT$</td>
<td>$TGCACT</td>
<td>T</td>
</tr>
<tr>
<td>GCATST</td>
<td>ACT$TGC</td>
<td>C</td>
</tr>
<tr>
<td>CACTTG</td>
<td>CACT$TG</td>
<td>G</td>
</tr>
<tr>
<td>ACT$TGC</td>
<td>CT$TGA</td>
<td>A</td>
</tr>
<tr>
<td>CT$TGA</td>
<td>GCATST</td>
<td>T</td>
</tr>
<tr>
<td>T$TGCAT</td>
<td>T$TGCAC</td>
<td>C</td>
</tr>
<tr>
<td>$TGCACT</td>
<td>TGCAT$</td>
<td>$</td>
</tr>
</tbody>
</table>

www.bcgsc.ca -> software

ABySS,
Trans-ABySS,
ABySS-Explorer,
Anchor

ABySS and Trans-ABySS Google Groups
ibirol@bcgsc.ca
sjackman@bcgsc.ca
ABySS Team:
Shaun Jackman
Tony Raymond
Rod Docking
Cydney Nielsen

Beetle Project:
Joerg Bohlmann
Chris Keeling

Trans-ABySS Team:
Readman Chiu
Karen Mungall
Gordon Robertson
Ka Ming Nip
Jenny Qian
Rong She
Lucas Swanson

GSC:
Sequencing Team
Library Core

Steven Jones
Marco Marra

Genome Canada
Genome British Columbia