ABySS

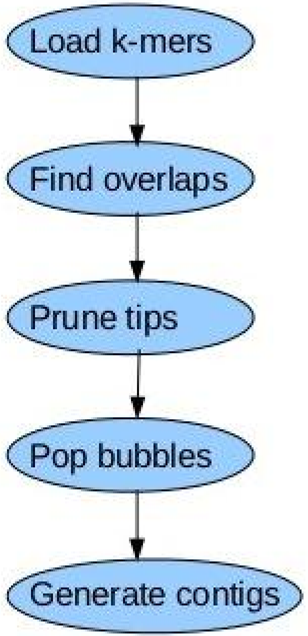
Assembly By Short Sequences

ABySS

- Developed at Canada’s Michael Smith Genome Sciences Centre
- Developed in response to memory demands of conventional DBG assembly methods
- Parallelizability
- Illumina recommended assembler for large genomes

Assembler Overview

- Break Reads into K-mers
- Find adjacency kmers
  - overlap by k-1 bases
- Generate De Bruijn graph
- Trim branches
- Pop bubbles
- Output Contigs

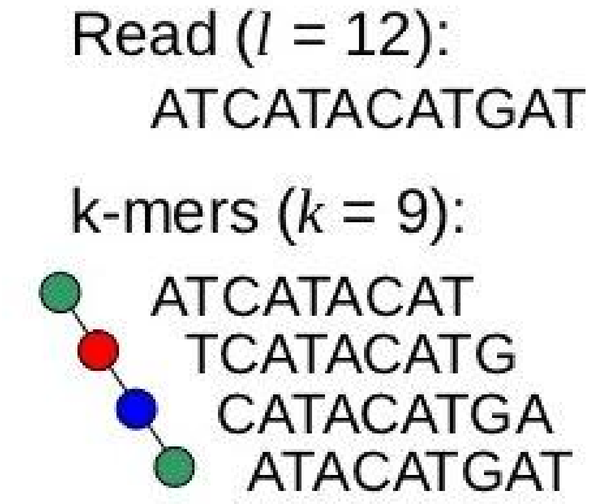


Loading K-mers

For each input read of length l,

(l - k + 1) k-mers are generated by sliding a window of length k over the read.

Each K-mer will be a Vertex in the De Bruijn graph and two adjacent K-mers are an edge of length k-1 in the graph.



K-mer Hash Table

"To distribute the de Bruijn graph over a network of computers we need to address two issues. First, the location of a given *k*-mer must be deterministically and efficiently computable from the sequence of the *k*-mer. Second, the adjacency information between *k*-mers must be stored in a manner that is independent of the actual location of the *k*-mer."

"A single *k*-mer, or vertex, can have up to eight edges—one for every possible one-base extension, {A, C, G, T}, in either direction. This information can be efficiently stored in 8 bits per *k*-mer, where one bit represents the presence or absence of each edge."

from **ABySS: A parallel assembler for short read sequence data**

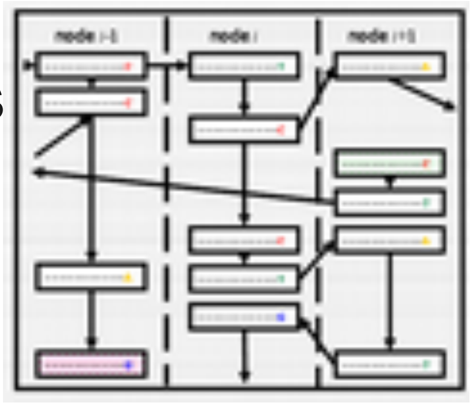
doi: 10.1101/gr.089532.108

K-mer Adjacency

Distribute the sequences over a cluster of computer nodes. The cluster node index of the k-mer is computed and the k-mer is assigned to this node for storage in a hash table.

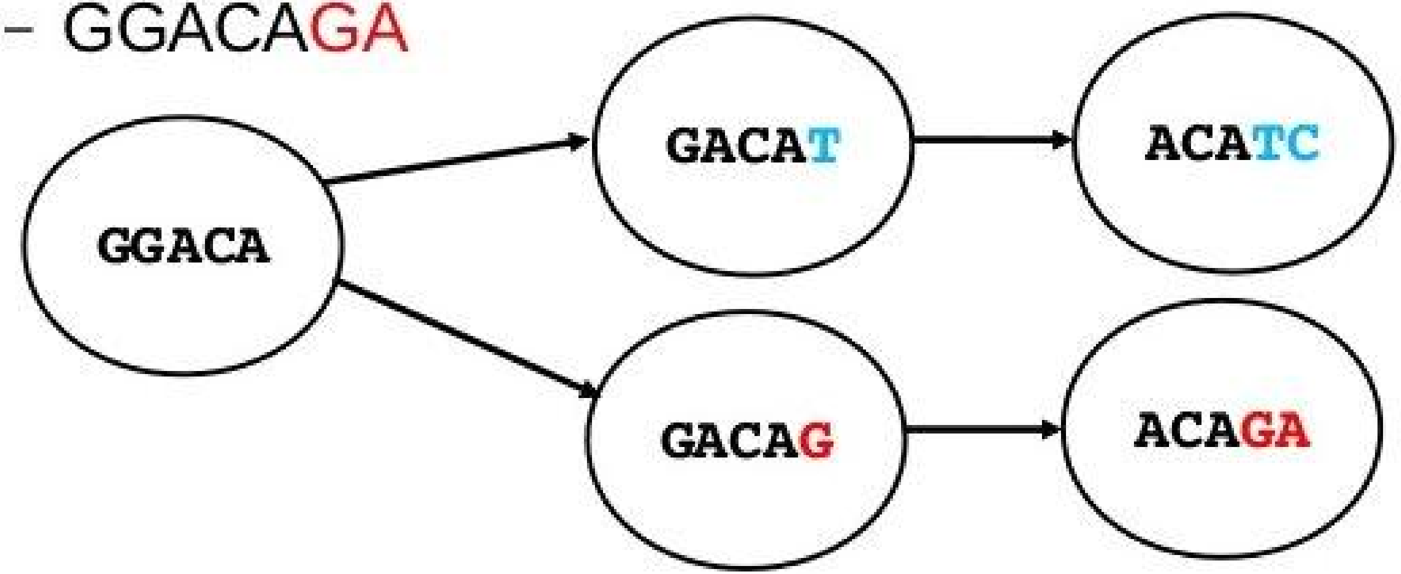
Each node announces the list of k-mers that it has to the nodes that hold their possible extensions.
Each node records if there are any extensions of the k-mers that it stores.

This forms the Adjacency information for k-mers over a distributed de Bruijn graph



De Bruijn graph

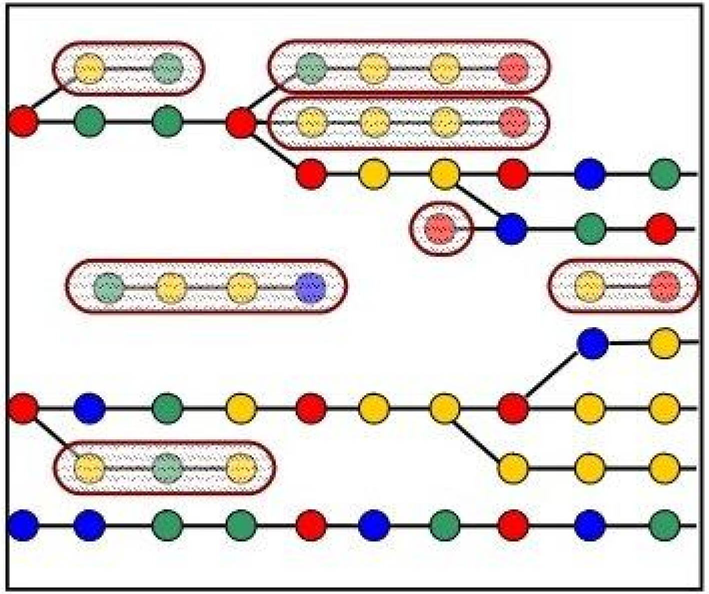
- GGACATC
- GGACAGA



Pruning

Certain Sequencing errors will cause “tips” to form in the graph.

ABySS “prunes” tips to avoid erroneous reads corrupting assembly.

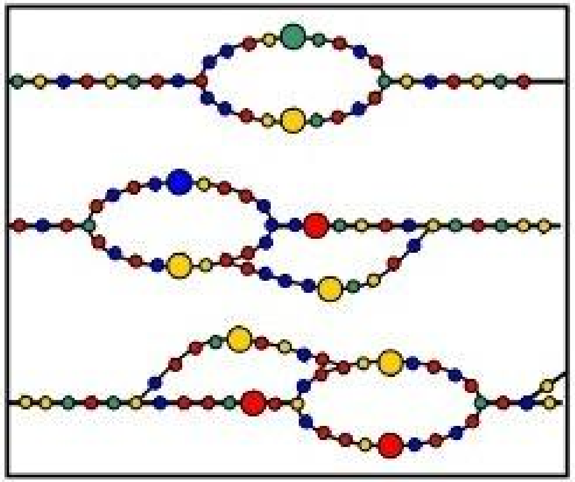


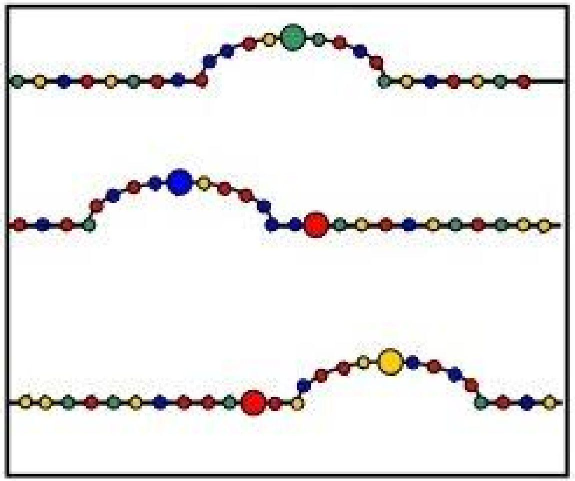
Popping

Genetic variance in sample generates bubbles.

Popping bubbles removes variant sequence from assembly.

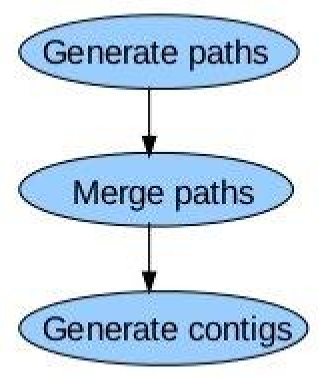
ABySS saves the variant data.





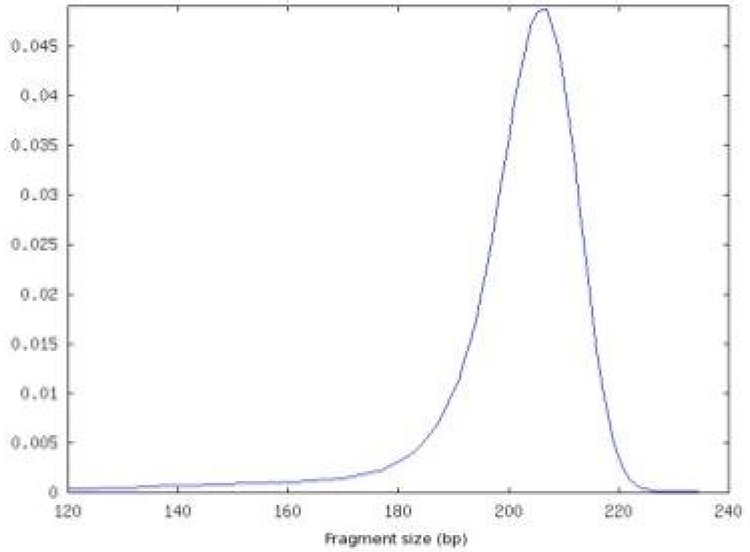
But wait there’s MORE!

- Find paths through the contig adjacency graph that agree with the distance estimates.
- Merge overlapping paths.
- Merge the contigs in these paths and output the FASTA file.

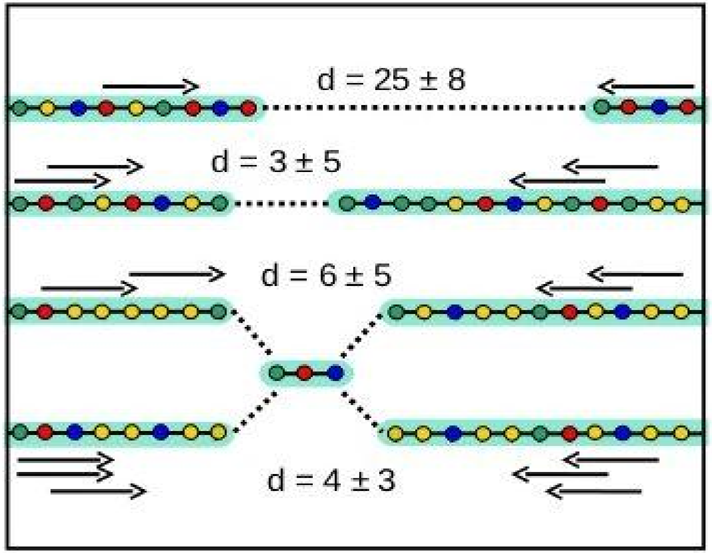


Paired Read Data is Cool

ParseAligns: Empirical fragment-size distribution



DistanceEst: Estimate distances between contigs



DistanceEst: Maximum Likelihood Estimator

1. Use the empirical paired- end size distribution.

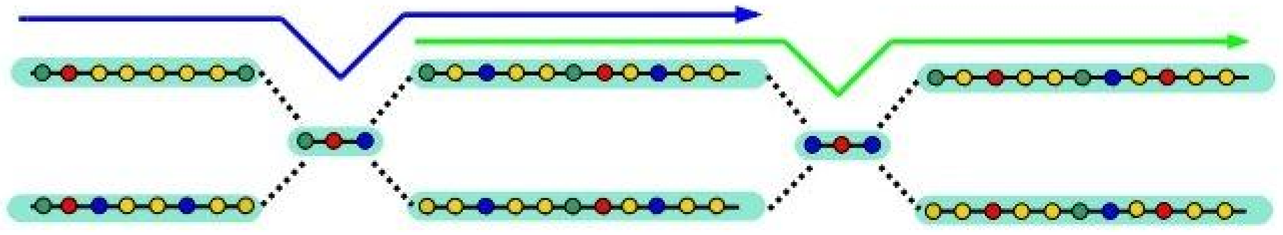
Likelihood is used when describing a function of a parameter given an *outcome*

2. Maximize the likelihood function.
3. Find the most likely distance between the two contigs.

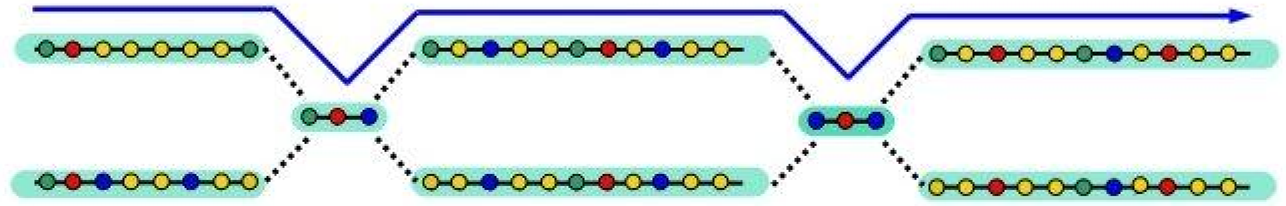
$$\mathcal{L}(\theta|x) = P(x|\theta). \qquad \mathcal{L}(\theta|x) = f_\theta(x), \qquad \mathcal{L}(\theta) = \prod_{i=1}^{n} f_\theta(x_i)$$

Merge Paths

SimpleGraph: Find consistent paths



MergePaths: Merge overlapping paths



User experience

Install, Run, Optimize, Parallelize

Installing

Dependencies:

- Google sparsehash: efficient hash implementation
- openmpi: enables parallel computing
  - --with-sge
- boost: collection of C++ libraries

Running

- Single Processor Version: Straight Forward
  - qsub slug.x.sh
  - embedded qsub options
  - exporting paths
  - abyss-pe [PARAMETERS]
- Parallel Processing Option: ...
  - specify PE, number of processes (np)
  - Sourcing issues? Administrative obstacles?

Parameters:

- Primary:
  - **name**: name of assembly
  - **k**: size of k-mer
  - if 1 library of pe data:
    - **in** = ‘reads1.fq reads2.fq’
  - else if multiple pe libs:
    - **lib** = ‘lib1 lib2’
    - **lib1** = ‘reads1.1.fq reads1.2.fq’
    - **lib2** = ‘reads2.1.fq reads2.2.fq’
  - else:
    - **se** = ‘reads.fq’

- Secondary:
  - **n**: min number of pairs required to join two contigs
  - **c**: mean k-mer coverage threshold
  - **q:** trim ends w/ bases lower than specified quality score
  - **np:** number of processes for mpi assembly
  - **mp**: mate-pair libraries

Convenience

- Pipeline organized via makefile: abyss-pe
  - ensures dependencies are generated
  - step-wise execution of Makefile enables easy troubleshooting at any point in pipeline
  - job can be stopped and resumed later
- tight integration of openmpi and sge
- auto generated assembly statistics
  - contig, scaffold metrics

Output overview

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

Test Run:

- S. cerevisiae paired end library
  - small tractable data set
  - ensure pipeline functions properly
  - provides an example of typical output

Parallelization

- distributed processing capabilities enable rapid assembly of large genomes
- reduces the effects of individual machine limitations

Using ABySS

What we did

The Plan

- Use all libraries, after preprocessing
    - (no error correction)
- Run for large range of k
    - Nice, easy syntax for setting this up
    - Big problem: Parallel version not working properly
- Determine best k retroactively
- Improve assembly

SeqPrep

- Two runs:
  - Adapter trimming only
  - Adapter trimming plus merging
- Kmergenie results: ???
- Future: Fastqc

Initial run

- edser
- k=55 (arbitrary)
- -j option: allows many jobs
  - didn’t work, did without
- used SW018 and SW019_S1 (couldn’t copy other files from campus rocks)

Initial run--Outcomes

- Parallel version not working
  - loses much of the benefit of ABySS
- Running basic version is easy
- Results: TBA

To Do List

- Get parallel versions working
- Finish data analysis (kmergenie, fastqc, etc)
- Do assemblies for many ks with all data
  - Including Lucigen data, new data
- Pick best assembly based on stats

Future ideas

- RNA-seq rescaffolding (with Trans-ABySS!)
- Meta-assembly