Sequence Assembly

BCB 504: Applied Bioinformatics

Matt Settles

University of Idaho
Bioinformatics and Computational Biology Program

March 18, 2013

Outline

Assembly

Shotgun Sequencing (Fred Sanger 1982) is the prevalent technique today

- Physically and randomly break the DNA into known sizes fragments.
- DNA sequencer reads the DNA fragments.
- Assembler reconstructs the original long sequence.

Assembly is challenging

- Data contains errors.
- DNA has repetitive sections called repeats.
- Gaps or un-sequenced fragments.
- Lots of data, computationally difficult.

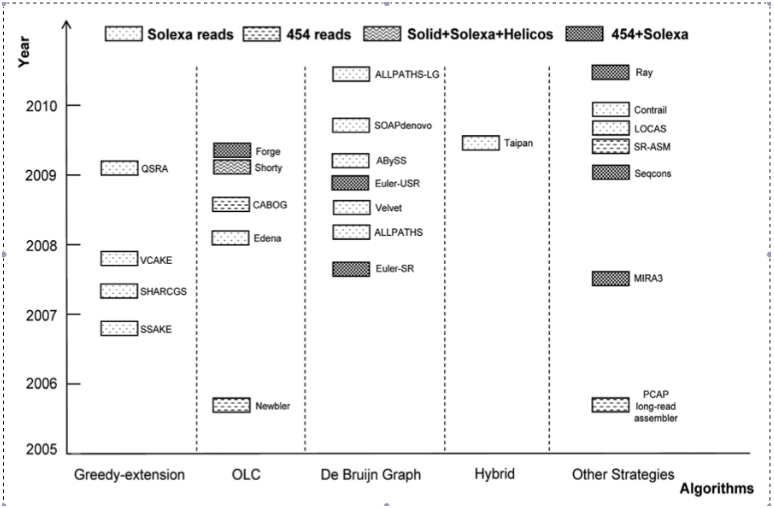
Basic Approach - de novo assembly

1. Sequence DNA to a specified depth.
2. Assemble reads into continuous sequence (aka contigs).
3. Order and orient contigs using additional information (aka scaffold).
   - Paired end (or mate pair reads).
   - Optical mapping (high-resolution restriction maps)
4. Finish (gap closure, resequence low quality regions)

Assembly Strategies

- Greedy-extension (old school)
- Overlap-layout-consensus (OLC) (small genomes)
- De Bruijn graph (large genomes)

Algorithms

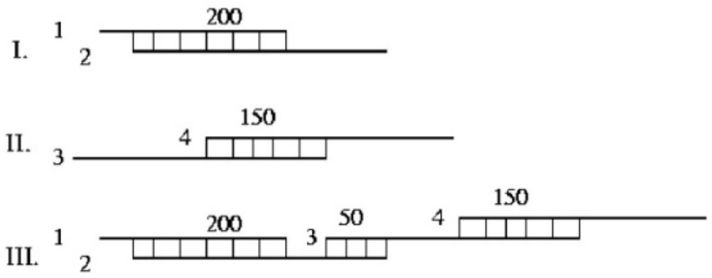

Year
Solexa reads
454 reads
Solid+Solexa+Helicos
454+Solexa
2010
2009
2008
2007
2006
2005
QSRA
VCAKE
SHARCGS
SSAKE
Forge
Shorty
CABOG
Edena
Newbler
ALLPATHS-LG
SOAPdenovo
ABySS
Euler-USR
Velvet
ALLPATHS
Euler-SR
Taipan
Ray
Contrail
LOCAS
SR-ASM
Seqcons
MIRA3
PCAP long-read assembler
Greedy-extension
OLC
De Bruijn Graph
Hybrid
Other Strategies
Algorithms

Greedy-extension

Simple but effective strategy in which the assembler greedily joins together the reads that are the most similar to each other (largest overlap), as long as they do not contradict the already contructed assembly. A major disadvantage of the simple greedy approach is

that because local information is considered at each step and not the global relationship between reads, the assembler can be easily confused by complex repeats, leading to mis-assemblies.

Greedy-extension (cont.)



Overlap-Layout-Consensus

The assembler starts by identifying all paris of reads that overlap sufficiently (by some predetermined criteria) and then organizes this information into a graph containing a node for every read and an edge between any pair of reads that overlap each other. The graph structure allows the development of complex assembly algorithms that can take into account the global relationship between the reads. Largest concern of OLC assemblers is the

computational complexity of the overlap computation.

Overlap-Layout-Consensus

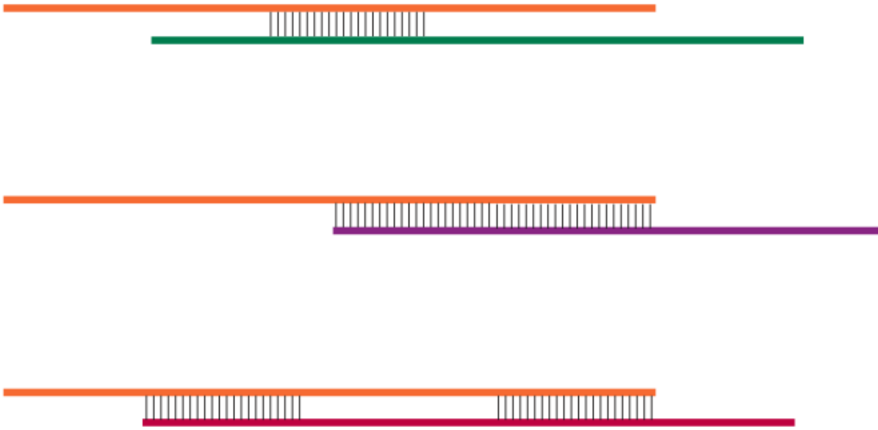
```
To be or not to
    e or not to be, th
         not to be, that is the q
                 e, that is the question.
To be or not to be, that is the question.
```

Overlap-Layout-Consensus (overlap)

overlap find all the significant matches between pairs of reads
complexity of the order of N x N
remember all significant matches
Smarter solution:

- Only n x coverage (e.g. 8) pairs are possible
- Build a table of k-mers contained in sequences (single pass through the genome)
- Generate the pairs from k-mer table (single pass through k-mer table)

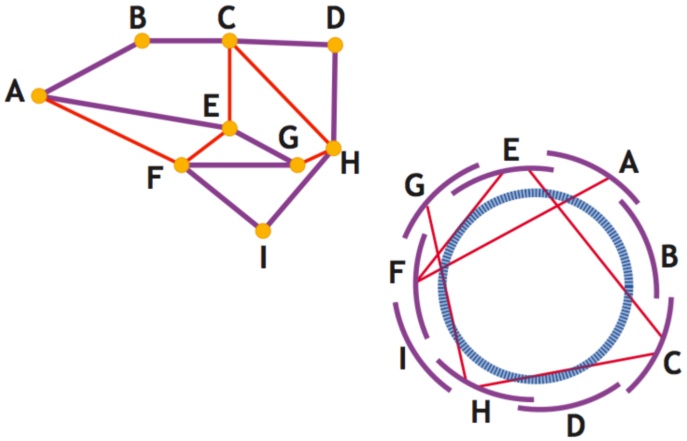
Overlap-Layout-Consensus (overlap)



Overlap-Layout-Consensus (layout)

layout parse the list of matches to satisfy simplest path through the list matching paired reads.
graph is simplified by removing redundant information.
Graph algorithms (Hamiltonian path,Eulerian path others) are then used to determine a layout (relative placement) of the reads along the genome

Overlap-Layout-Consensus (layout)



Overlap-Layout-Consensus (consensus)

consensus Derive a single consensus sequence for each set of contiguated reads (contig) following rules to deal with heterozygosity, errors and anomalies.
Builds an alignment of all the reads covering the genome and infers, as a consensus of the aligned reads, the original sequence of the genome being assembled.

De Bruijn Graphs

Assembling hundreds of millions of reads (and very short reads)

Problem with Overlap-Layout-Consensus
The layout problem is formulated as the Shortest Superstring Problem, but the major difficulty is that there is no efficient algorithm for the solution of the layout problem.

Heuristics are used to overcome this problem, but will often induce errors. Enter De Bruijn Assemblers

De Bruijn Graphs

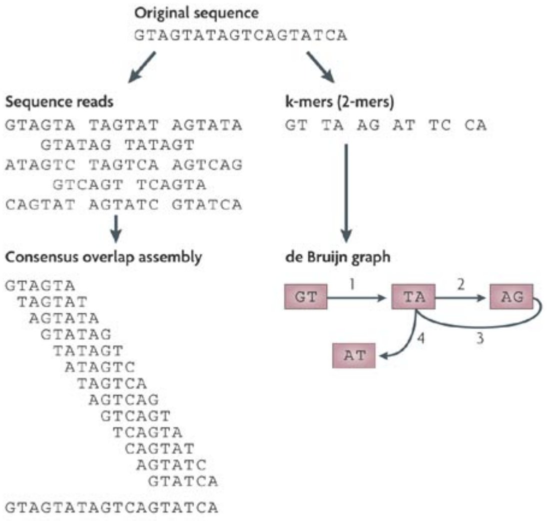
De Bruijn graph assemblers model the relationship betwee exact substrings of length k extracted from the input reads. Similarly to the OLC approach, the nodes in the graph represent k-mers, and the edges indicate that the adjacent k-mers overlap by exactly k-1 letters. Whereas the reads themselves are not directly modelled in this paradigm (as they are in OLC), they are implicitely represented as paths through the de Bruijn graph. Most de Bruijn assemblers

us the read information to refine the graph structure and to remove graph patterns that are not consistant with the reads. Since the de Bruijn approach is based on exact matches, error correction approaches are crucial for achieving high quality assemblies.

De Bruijn Graphs

- split reads into k-mers
  - k < read length (usually << read length), but > 19
  - K must also be odd
- build De Bruijn graph of k-mers
  - nodes are k-mers and edges indicate overlaps (reads can be split)
- deconvolute graph to derive assembly
  - the Eulerian cycle problem - visit every edge once (simpler to solve)

De Bruijn Graphs



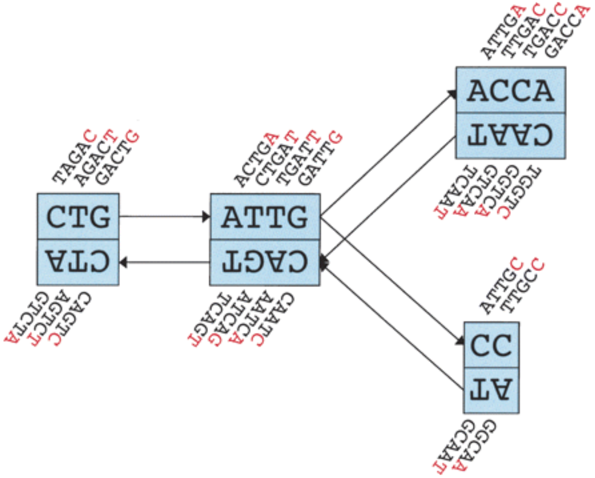
De Bruijn Graphs

In the de Bruijn graph, each node N represents a series of overlapping k-mers. Adjacent k-mers overlap by k-1 nucleotides.

The marginal information contained by a k-mer is its last nucleotide. The seuqence of those final nucleotides is called the sequences of the node ($s(N)$). Each node N is attached to a twin node N_c which represenet the reverse compliment of the k-mers. N and N_c for a BLOCK.

BLOCKS are joined by ARCS.
The last k-mer of an arc's origin node overlaps with the first of its destination node.
Reads are mapped as "paths" traversing the graph.

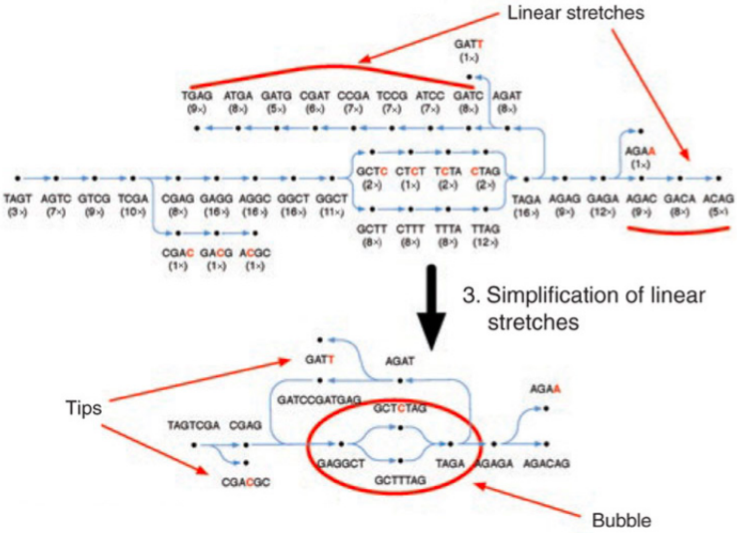
De Bruijn Graphs



De Bruijn Graphs

Simplification
After constructing the graph, it is generally possible to simplify it without any loss of information. Whenever a node A has only one outgoing arc that points to another node B that has only one ingoingarc, the nodes (and their twins) can be merged.

De Bruijn Graphs



De Bruijn Graphs - error structures I

Error removal

Erroneous data create three types of structures:

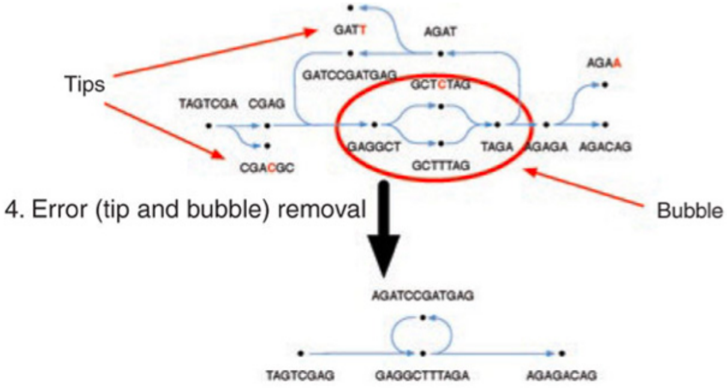
- "tips" due to errors at the edges of reads.
  - A "tip: is a chain of nodes that is disconnected at one end. Removing these tips is a straightforward task. Discarding this information results in only local effects and no connectivity is disrupted.
- "bulges" or bubbles due to internal read errors or to nearby tips connecting.
  - Two paths are redundant if they start and end at the same nodes and contain similar sequences.

De Bruijn Graphs - error structures

Error removal (cont)

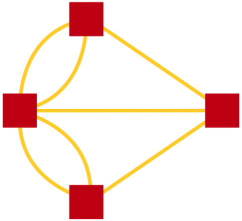
- erroneous connections due to cloning errors or to distant merging tips.
  - After the above corrections, genuine short nodes that cannot be simplified correspond to low-complexity sequences that are generally present multiple times in the genome. Their overall coverage is therefore proportionally higher than expected. This means that with a high probability, any low coverage node left after tip and bulge resolution is a chimeric connection, due to spurious overlaps, created by experimental errors.

De Bruijn Graphs - error structures



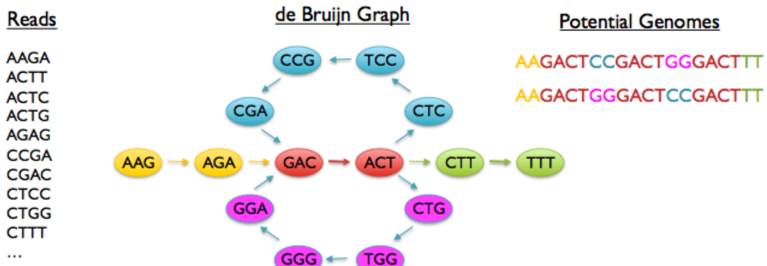
De Bruijn Graphs - deconvolute graph

deconvolute graph to derive assembly
-the Eulerian cycle problem, find a path that visits every edge once



tends to be exponential BUT can efficiently traverse the graph using the BEST theorem

De Bruijn Graphs - deconvolute graph



De Bruijn graph assembly and traversal is compute intensive:
Building a graph for the human genome from real data requires $>$ 3 billion nodes and $>$ 10 billion edges

Assembly Parameter Considerations

k-mer choice

Longer k-mers bring you more specificity (i.e. less spurious overlaps) but lowers coverage, so there's a sweet spot to be found with time and experience.

Velvet suggests to think in terms of "k-mer coverage", i.e. how many times has a k-mer been seen among the reads. The relation between k-mer coverage (C_k) and nucleotide coverage (C) is $C_k = C * (l - k + 1)/L$, where k is the k-mer length (hash length), and L is the read length.

Velvet suggests that the k-mer coverage should be above 10 and If C_k is above 30, you might be "wasting" coverage (20-30 is supposed to be ideal). k-mer choice is going to have the largest influence in the results, so its often best to try many.

Resources

- http://seqanswers.com
- http://seqanswers.com/wiki/SEQanswers