Introduction to genome assembly

CMSC423

Many slides courtesy of Ben Langmead and Carl Kingsford
Sec-gen Sequencing
Sec-gen Sequencing

Fragmentation is random, i.e., not equal-sized (but hard to draw)
Sec-gen Sequencing
**Second-Generation Sequencing**

- “Ultra high throughput” DNA sequencing
- 6 gigabases / day vs.
- 3 gigabases / 13 years (human genome project, more or less)
- 200 bp long reads
From reads to evidence
From reads to evidence

I. **de novo**

Assume nothing! - let reads tell us everything

Reads with overlapping sequence probably originate from overlapping portions of the subject genome

Encode overlap relationships as a graph

The full genome sequence is a “tour” of the graph

Source: De Novo Assembly Using Illumina Reads. Illumina. 2010

What we’ll cover

• Genome assembly as graph problems

• Two representations:
  • Overlap graph
    • How much sequencing required for assembly
  • DeBruijn graph

• How to get assemblies from solutions to graph problems
Overlap graph:
Nodes = reads
Edges = overlaps

Given overlap graph, how can we find a good candidate assembly?
Overlap graph:
Nodes = reads
Edges = overlaps

Given overlap graph, how can we find a good candidate assembly?
Overlap Graph

Overlap graph:
Nodes = reads
Edges = overlaps

Given overlap graph, how can we find a good candidate assembly?

Hamiltonian Path (aka Traveling Salesman Path): visit every node in
the graph exactly once.
Hamiltonian Path

- Motivation: Every read must be used in exactly one place in the genome.

- Hamiltonian Path is NP-hard.

- Though good solvers exist, they can’t operate on the millions of reads from a sequencing project.

- Solution: greedy walk along the graph.

Many copies of the DNA

Shear it, randomly breaking them into many small pieces, read ends of each:

Assemble into original genome:
Lander-Waterman Statistics

How many reads do we need to be sure we cover the whole genome?

An *island* is a contiguous group of reads that are connected by overlaps of length $\geq \theta L$.
(Various colors above)

Want: Expression for expected # of islands given $N, g, L, \theta$. 
Expected # of Islands

\[ \lambda := \frac{N}{g} = \text{probability a read starts at a given position} \]
(assuming random sampling)

\[ \Pr(k \text{ reads start in an interval of length } x) \]
\[ x \text{ trials, want } k \text{ “successes,” small probability } \lambda \text{ of success} \]
Expected # of successes = \( \lambda x \)
Poisson approximation to binomial distribution:

\[ \Pr(k \text{ reads in length } x) = e^{-\lambda x} \frac{(\lambda x)^k}{k!} \]

Expected # of islands = \( N \times \Pr(\text{read is at rightmost end of island}) \)

\[ \frac{(1-\theta)L}{\theta L} = N \times \Pr(0 \text{ reads start in } (1-\theta)L) \]
\[ = Ne^{-\lambda(1-\theta)L} \frac{(\lambda(1 - \theta)L)^0}{0!} \]
\[ = Ne^{-\lambda(1-\theta)L} \]
\[ = Ne^{-(1-\theta)LN/g} \] ← \( LN/g \) is called the coverage c.
Expected # of Islands, 2

Rewrite to depend more directly on the things we can control: $c$ and $\theta$

\[
\text{Expected # of islands} = Ne^{-(1-\theta)LN/g} = Ne^{-(1-\theta)c} = \frac{L/g}{L/g} Ne^{-(1-\theta)c} = \frac{L/g}{L/g} \cdot \frac{g}{L} ce^{-(1-\theta)c}
\]
Assembly via Eulerian Path
**de Bruijn graph**: nodes represent kmers, edges connect k-mers that are known to follow each other based on an observed read.

Can have > 1 edge between nodes.
Example bacterial de Bruijn graph

Paths with no branches compressed into a single node

Eulerian path = use every edge exactly once.

With perfect data, the genome can be reconstructed by some Eulerian path through this graph.
Let $dG(s)$ be the de Bruijn graph of string $s$. Then $s$ corresponds to some Eulerian path in $dG(s)$.

A directed graph has an Eulerian path if and only if:

- One node has one more edge leaving it than entering
- One node has one more edge entering than leaving
- All other nodes have the same number of edges entering and leaving

How can we find such a path?
A directed graph has an Eulerian cycle if and only if:

- All nodes have the same number of edges entering and leaving.
Eulerian Path Algorithm

Connect node with out-degree < in-degree to node with out-degree < in-degree. So that we will have an Eulerian cycle.

Walk from some arbitrary node $u$ until you return to $u$, creating a doubly liked list of the path you visit.

**Repeat** until all edges used:
- Start from some node $w$ on the current tour with unused edges*.
- Walk along unused edges until you return to $w$, inserting the visited nodes after $w$ into the current tour list.

*How can find such a node quickly?

Why will you return to $u$?
Connect node with out-degree < in-degree to node with out-degree < in-degree. So that we will have an Eulerian cycle.

Walk from some arbitrary node $u$ until you return to $u$, creating a doubly liked list of the path you visit.

**Repeat** until all edges used:
- Start from some node $w$ on the current tour with unused edges.
- Walk along unused edges until you return to $w$, inserting the visited nodes after $w$ into the current tour list.
The Problem with Eulerian Paths

There are typically an astronomical number of possible Eulerian tours with perfect data.

Adding back constraints to limit # of tours leads to a NP-hard problem.

With imperfect data, there are usually NO Eulerian tours.

Aside: counting # of Eulerian tours in a directed graph is easy, but in an undirected graph is #P-complete (hard).

(Kingsford, Schatz, Pop, 2010)
Mate Pairs

Mate pair: 2 reads, of opposite orientation, separated by an approximately known distance

⇒ long range information
TAATGCCATGGGATGTT
Gapped Genome Path String Problem

Reconstruct a string from a sequence of (k, d)-mers corresponding to a path in a paired de Bruijn graph.

Given: A sequence of (k, d)-mers (a₁|b₁), ..., (aₙ|bₙ) such that Suffix(aᵢ|bᵢ) = Prefix(aᵢ₊₁|bᵢ₊₁) for all i from 1 to n-1.

Return: A string Text where the i-th k-mer in Text is equal to Suffix(aᵢ|bᵢ) for all i from 1 to n, if such a string exists.
References

• http://www.cbcb.umd.edu/research/assembly_primer

• http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2874646/

• http://www.math.ucsd.edu/~gptesler/186/slides/shotgun_f13-handout.pdf

• http://www.biomedcentral.com/1471-2105/11/21/abstract