Sequence Assembly

Fall 2016
BMI/CS 576

www.biostat.wisc.edu/bmi576/
Colin Dewey

colin.dewey@wisc.edu
The sequencing problem

• We want to determine the identity of the base pairs that make up:
  – A single large molecule of DNA
  – The genome of a single cell
  – The genome of an individual organism
  – The genome of a species

• But we can’t (currently) “read” off the sequence of an entire molecule all at once
The strategy: substrings

• We *do* have the ability to read or detect *short* pieces (substrings) of DNA
  – Sanger sequencing: 500-700 bp/read
  – Hybridization arrays: 8-30bp/probe
  – Latest technologies:
    • 454 Genome Sequencer FLX: 250-600 bp/read
    • Illumina Genome Analyzer: 35-300 bp/read
    • Pacific Biosciences: ~10,000 bp/read
Sanger sequencing

• Classic sequencing technique: “Chain-termination method”

• Replication terminated by inclusion of dideoxynucleotide (ddNTP)
Sequencing gels

- Run replication in four separate test tubes
  - Each with one of some concentration of either ddATP, ddTTP, ddGTP, or ddCTP
- Depending on when ddNTP is included, different length fragments are synthesized
- Fragments separated by length with electrophoresis gel
- Sequence can be read from bands on gel
Universal DNA arrays

• Array with all possible oligonucleotides (short DNA sequence) of a certain length as probes
• Sample is labeled and then washed over array
• Hybridization is detected from labels
Reading a DNA array
Latest technologies

• 454:
  – “Sequencing by synthesis”
  – Light emitted and detected on addition of a nucleotide by polymerase
  – 400-600 Mb / 10 hour run

• Illumina
  – Also “sequencing by synthesis”
  – ~100 Gb/day on one machine
  – Uses fluorescently-labeled reversible nucleotide terminators
  – Like Sanger, but detects added nucleotides with laser after each step
Latest technologies

• Pacific Biosciences:
  – “Sequencing by synthesis”
  – Single molecule sequencing
  – Detects addition of single fluorescently-labeled nucleotides by an immobilized DNA polymerase
  – Real-time: reads bases at the rate of DNA polymerase
  – 4 hours for sequencing with reads up to 60kb long
  – [video](#)
Oxford Nanopore

- Emerging technology
- Pocket-sized
- High error rate
- Currently in “community” program
Shotgun Sequencing Fragment Assembly

Multiple copies of sample DNA

Randomly fragment DNA

Sequence sample of fragments

Assemble reads
Two sequencing paradigms

1. Fragment assembly
   - For technologies that produce “reads”
     • Sanger, 454, Illumina, etc.

2. Spectral assembly
   - For technologies that produce “spectra”
     • Universal DNA arrays
   - Read data can also be “converted” to spectra

The two paradigms are actually closely related
The fragment assembly problem

• Given: A set of reads (strings) \( \{s_1, s_2, \ldots, s_n\} \)

• Do: Determine a large string \( s \) that “best explains” the reads

• What do we mean by “best explains”? 

• What assumptions might we require?
Shortest superstring problem

• Objective: Find a string $s$ such that
  – all reads $s_1, s_2, \ldots, s_n$ are substrings of $s$
  – $s$ is as short as possible

• Assumptions:
  – Reads are 100% accurate
  – Identical reads must come from the same location on the genome
  – “best” = “simplest”
Shortest superstring example

• Reads:
  \{ACG, CGA, CGC, CGT, GAC, GCG, GTA, TCG\}

• Shortest superstring (length 10)

  \textbf{TCGACGCGTA}
  TCG
  CGA
  GAC
  ACG
  CGC
  GCG
  CGT
  GTA
Algorithms for shortest substring problem

• This problem turns out to be \( NP \)-complete
• Simple \textit{greedy} strategy:
  while \# strings > 1 do
    merge two strings with maximum overlap
  loop
• Conjectured to give string with
  length \( \leq 2 \times \) minimum length
• “2-approximation”
• Other algorithms will require \textit{graph theory}...
Graph Basics

• A graph \((G)\) consists of vertices \((V)\) and edges \((E)\)
  \[G = (V,E)\]

• Edges can either be *directed* (*directed graphs*)

• or *undirected* (*undirected graphs*)
Vertex degrees

• The *degree* of a vertex: the # of edges incident to that vertex

• For directed graphs, we also have the notion of
  – *indegree*: The number incoming edges
  – *outdegree*: The number of outgoing edges

\[
\begin{align*}
\text{degree}(v_2) &= 3 \\
\text{indegree}(v_2) &= 1 \\
\text{outdegree}(v_2) &= 2
\end{align*}
\]
Overlap graph

• For a set of sequence reads $S$, construct a directed weighted graph $G = (V,E,w)$
  – with one vertex per read ($v_i$ corresponds to $s_i$)
  – edges between all vertices (a complete graph)
  – $w(v_i,v_j) = overlap(s_i,s_j) = \text{length of longest suffix of } s_i \text{ that is a prefix of } s_j$
Overlap graph example

- Let $S = \{\text{AGA, GAT, TCG, GAG}\}$
Assembly as Hamiltonian Path

- *Hamiltonian Path*: path through graph that visits each vertex exactly once

Path: AGAGATCG
Shortest superstring as TSP

• minimize superstring length ➔ minimize hamiltonian path length in overlap graph with edge weights negated

Path: GAGATCG
Path length: -5
String length: 7

• This is essentially the Traveling Salesman Problem (also NP-complete)
The Greedy Algorithm

• Let $G$ be a graph with fragments as vertices, and no edges to start

• Create a queue, $Q$, of overlap edges, with edges in order of increasing weight

• While $G$ is disconnected
  – Pop the next possible edge $e = (u,v)$ off of $Q$
  – If $\text{outdegree}(u) = 0$ and $\text{indegree}(v) = 0$ and $e$ does not create a cycle
    • Add $e$ to $G$
Greedy Algorithms

• **Definition**: An algorithm that always takes the best immediate, or local, solution while finding an answer.

• Greedy algorithms find the overall, or globally, optimal solution for some optimization problems, but may find less-than-optimal solutions for some instances of other problems.

Greedy Algorithm Examples

• Kruskal’s Algorithm for Minimum Spanning Tree
  – *Minimum spanning tree*: a set of n-1 edges that connects a graph of n vertices and that has minimal total weight
  – *Kruskal’s algorithm* adds the edge that connects two components with the smallest weight at each step
    • Proven to give an optimal solution

• Traveling Salesman Problem
  – Greedy algorithm chooses to visit closest vertex at each step
  – Can give far-from-optimal answers
Simplifications of overlap graph

- Require minimum length for overlap
- Linear chain compression
- Transitive edge removal
Sequencing by Hybridization (SBH)

• SBH array has probes for all possible $k$-mers
• For a given DNA sample, array tells us whether each $k$-mer is *PRESENT* or *ABSENT* in the sample
• The set of all $k$-mers present in a string $s$ is called its *spectrum*
• Example:
  - $s = \text{ACTGATGCGAT}$
  - $\text{spectrum}(s, 3) = \{\text{ACT, ATG, CAT, CTG, GAT, GCA, TGA, TGC}\}$
Example DNA Array

<table>
<thead>
<tr>
<th></th>
<th>AA</th>
<th>AC</th>
<th>AG</th>
<th>AT</th>
<th>CA</th>
<th>CC</th>
<th>CG</th>
<th>CT</th>
<th>GA</th>
<th>GC</th>
<th>GG</th>
<th>GT</th>
<th>TA</th>
<th>TC</th>
<th>TG</th>
<th>TT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>AG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>GA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>GG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XX</td>
</tr>
</tbody>
</table>

Sample: ACTGATGCAT
Spectrum (k=4): {ACTG, ATGC, CTGA, GATG, GCAT, TGAT, TGCA}
SBH Problem

• Given: A set $S$ of $k$-mers
• Do: Find a string $s$, such that $\text{spectrum}(s,k) = S$

$\{\text{ACT, ATG, CAT, CTG, GAT, GCA, TGA, TGC}\}$
SBH as Eulerian path

• Could use Hamiltonian path approach, but not useful due to $NP$-completeness
• Instead, use *Eulerian* path approach

*Eulerian path*: A path through a graph that traverses every edge exactly once

• Construct graph with all $(k-1)$-mers as vertices
• For each $k$-mer in spectrum, add edge from vertex representing first $k-1$ characters to vertex representing last $k-1$ characters
Properties of Eulerian graphs

• It will be easier to consider *Eulerian cycles*: Eulerian paths that form a cycle
• Graphs that have an *Eulerian cycle* are simply called *Eulerian*
• **Theorem**: A connected directed graph is *Eulerian* if and only if each of its vertices are *balanced*
• A vertex $v$ is *balanced* if $\text{indegree}(v) = \text{outdegree}(v)$
• There is a polynomial-time algorithm for finding Eulerian cycles!
Seven Bridges of Königsberg

Euler answered the question: “Is there a walk through the city that traverses each bridge exactly once?”
Eulerian cycle algorithm

• Start at any vertex $v$, traverse unused edges until returning to $v$

• While the cycle is not Eulerian
  – Pick a vertex $w$ along the cycle for which there are untraversed outgoing edges
  – Traverse unused edges until ending up back at $w$
  – Join two cycles into one cycle
Joining cycles
Eulerian Path -> Eulerian Cycle

• If a graph has an Eulerian Path starting at $s$ and ending at $t$ then
  
  – All vertices must be balanced, except for $s$ and $t$ which may have $|\text{indegree}(v) - \text{outdegree}(v)| = 1$
  
  – If $s$ and $t$ are not balanced, add an edge between them to balance

• Graph now has an Eulerian cycle which can be converted to an Eulerian path by removal of the added edge
SBH graph example

{ACT, ATG, CAT, CTG, GAT, GCA, TGA, TGC}
SBH difficulties

• In practice, sequencing by hybridization is hard
  – Arrays are often inaccurate -> incorrect spectra
    • False positives/negatives
  – Need long probes to deal with repetitive sequence
    • But the number of probes needed is exponential in the length of the probes!
    • There is a limit to the number of probes per array (currently between 1-10 million probes / array)
K-mer spectrum approach with read data (de Bruijn approach)

• Generate spectrum from set of all $k$-mers contained within reads
• Choose $k$ to be small enough such that the majority of the genome’s $k$-mers will be found within the reads
• Particularly useful for short-read data, such as that produced by Illumina
• Made popular by methods such as Euler and Velvet
Difficulties with de Bruijn approach

• Not all $k$-mers may be contained within the reads even if reads completely cover the genome
• DNA repeats result in $k$-mers that are present in multiple copies across the genome
• Reads often have sequencing errors!
Fragment assembly challenges

• Read errors
  – Complicates computing read overlaps

• Repeats
  – Roughly half of the human genome is composed of repetitive elements
  – Repetitive elements can be long (1000s of bp)
  – Human genome
    • 1 million Alu repeats (~300 bp)
    • 200,000 LINE repeats (~1000 bp)
Overlap-Layout-Consensus

• Most common assembler strategy for long reads

1. *Overlap*: Find all significant overlaps between reads, allowing for errors

2. *Layout*: Determine path through overlapping reads representing assembled sequence

3. *Consensus*: Correct for errors in reads using layout
Consensus

Layout

GTATCGTAGCTGACTGCGCTGC
ATCGTCTCGTAGCTGACTGCGCTGC
ATCGTATCGAATCGTAG
TGACTGCGCTGCATCGTATCGTATC

Consensus

TGACTGCGCTGCATCGTATCGTATC
ATCGTAGCTGACTGCGCTGC
Whole Genome Sequencing

• Two main strategies:

1. Clone-by-clone mapping
   • Fragment genome into large pieces, insert into BACs (Bacterial Artificial Chromosomes)
   • Choose *tiling set* of BACs: overlapping set that covers entire genome
   • Shotgun sequence the BACs

2. Whole-genome shotgun
   • Shotgun sequence the entire genome at once
Assembly in practice

• Assembly methods used in practice are complex
  – But generally follow one of the two approaches
    • Reads as vertices
    • Reads as edges (or paths of edges)

• Assemblies do not typically give whole chromosomes
  – Instead gives a set of “contigs”
  – contig: contiguous piece of sequence from overlapping reads
  – contigs can be ordered into scaffolds with extra information (e.g., paired end reads)
Cloning and Paired-end reads

1. **DNA fragment**
2. **vector**
3. **Insert fragment into vector**
4. **transform bacteria with vector and grow**
5. **...**
6. **Sequence ends of insert using flanking primers**
Paired-end read advantages

• *Scaffolding*: layout of adjacent, but not overlapping, *contigs*

• *Gap filling*:
Sequence assembly summary

• Two general algorithmic strategies
  – Overlap graph hamiltonian paths
  – Eulerian paths in k-mer graphs

• Biggest challenge
  – Repeats!
    • Large genomes have a lot of repetitive sequence

• Sequencing strategies
  – Clone-by-clone: break the problem into smaller pieces which have fewer repeats
  – Whole-genome shotgun: use paired-end reads to assemble around and inside repeats