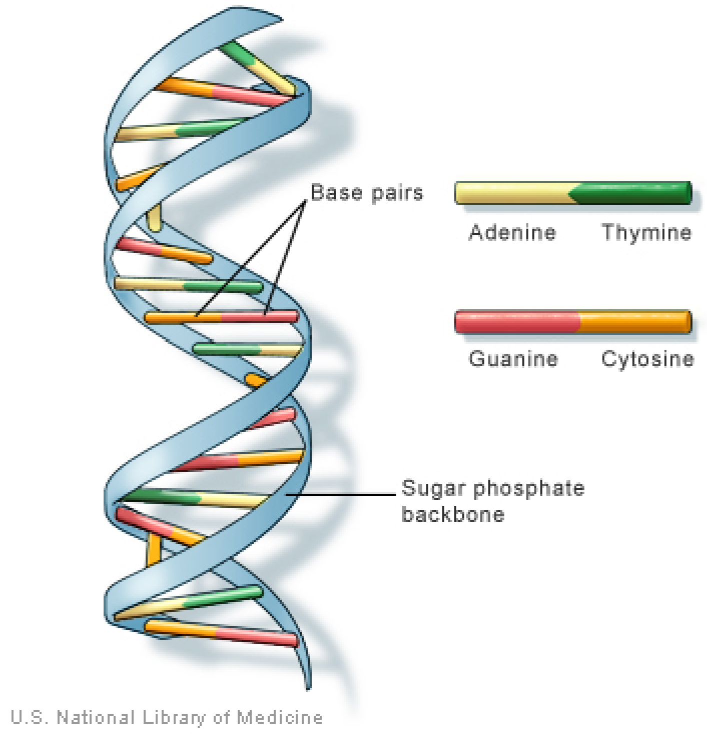
Reducing Genome Assembly Complexity with Optical Maps

De Bruijn graphs provide a framework for genome assembly, where each Eulerian trail through the graph corresponds to a possible reconstruction of the genome. The assembly problem is complicated by genomic repeats, which introduce many alternative possible reconstructions and increase the complexity of the de Bruijn graph. Optical maps provide an ordered listing of restriction fragment sizes for a particular enzyme across an entire chromosome, and therefore give long range information that can be useful in resolving genomic repeats. The algorithms presented here align contigs to an optical map and use this information to find paths through the assembly graph that can resolve genomic repeats, thereby reducing the assembly graph complexity.

• Lee Mendelowitz
Lmendelo@math.umd.edu

• Advisor: Mihai Pop
mpop@umiacs.umd.edu
Computer Science Department
Center for Bioinformatics and Computational Biology

DNA Sequences



- Four DNA Bases: Adenine (A), Guanine (G), Cytosine (C), Thymine (T)
- Complementary Base Pairs (A-T) & (G-C)
- DNA is double stranded with antiparallel strands
- Synthesized in 5' to 3' direction
- Example:
  - (3') A T G C T C (5')
  - (5') T A C G A G (3')
- A **genome** is an organism's total DNA sequence, chromosome by chromosome.

DNA Sequencing

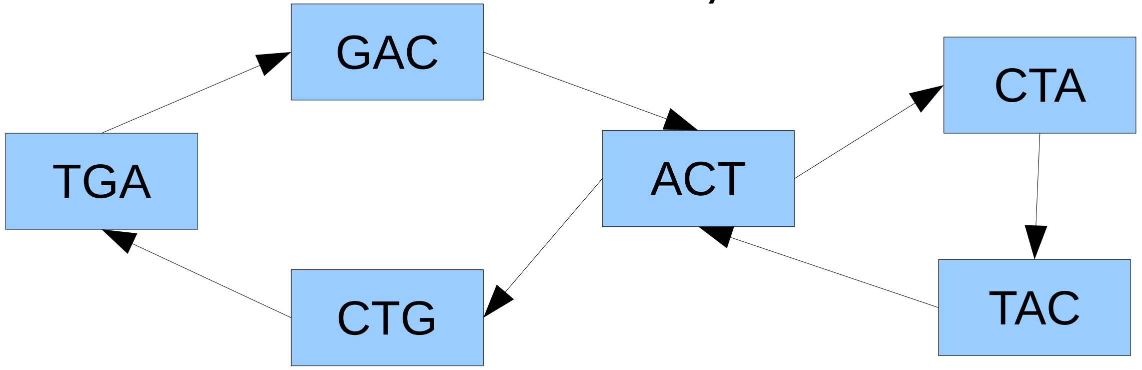
- DNA is extracted from a biological sample and randomly sheared into fragments. The DNA sequences at the end of the fragments are determined experimentally, producing a set of **reads.**

Sanger Sequencing	Second Generation Sequencing
~ 1975	~ 2005
Long Reads (1000-2000 bp)	Short reads (35 – 400 bp)
High Accuracy	Error Prone
Low Throughput	High Throughput
Low Coverage	High Coverage

- **Genome Assembly**: Using experimentally obtained reads, produce a single DNA sequence for each chromosome of the organism(s) whose DNA is sequenced.
  - In practice, an assembler cannot assemble an entire chromosome. Instead, it outputs a set of contiguous DNA sequences known as **contigs**

Genome Assembly with de Bruijn Graphs

- Create a vertex for each unique substring of length *K* (*k*-mer) from each read
- Draw a directed edge from vertex A to vertex B if *k*-mer B immediately follows *k*-mer A in a read.
  - Example: 'ACTACTGACT...' , K = 3



•A potential reconstruction of the genome corresponds to an **Eulerian tour** through the graph that visits each edge exactly once.

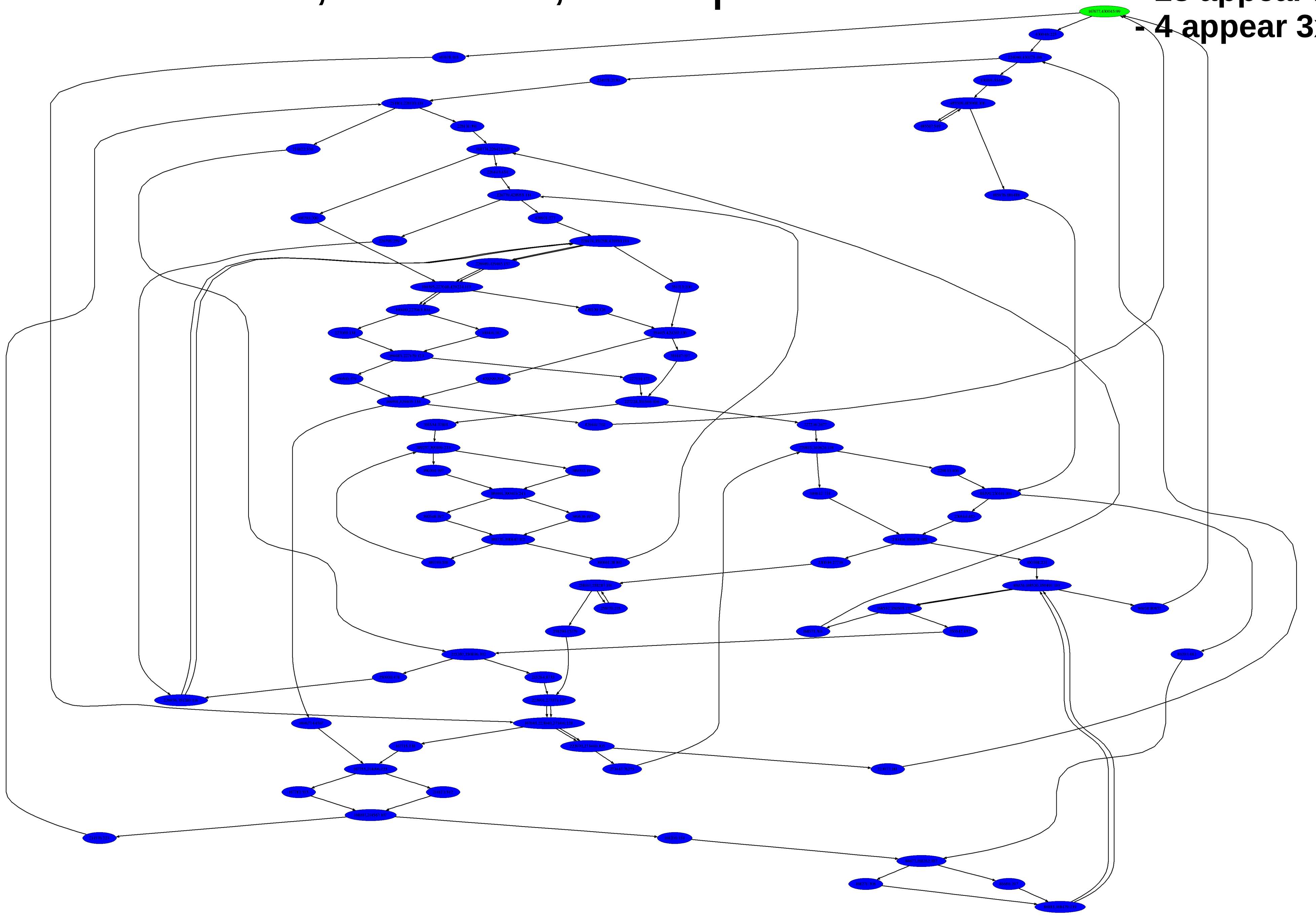
•Even with perfect data, **genome reconstruction with a de Bruijn graph is non-trivial due to genomic repeats**

- The number of possible genome reconstructions is **exponential in the number of repeats**

de Bruijn Graph, Mycoplasma genitalium

K=100, G = 580,076 bp

120 edges
84 vertices
- 52 appear 1x
- 28 appear 2x
- 4 appear 3x



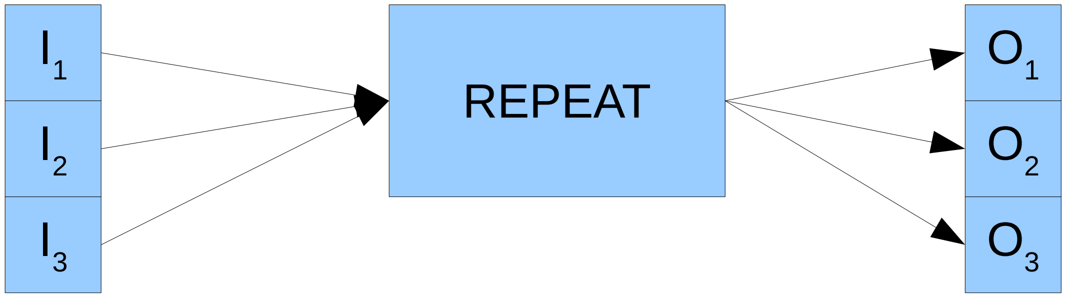
Assembly Graph Complexity

Let $d^+(v)$ be the incoming degree and $d^-(v)$ the outgoing degree of vertex v. For a given repeat vertex v with $d^+(v) = d^-(v) = a$, complexity of the vertex, $C(v)$, is defined to be equal to the number of targeted experiments necessary to match each incoming edge with each outgoing edge:

$$C(v) = \sum_{i=2}^{a} i = \frac{a(a+1)}{2} - 1$$

The total finishing complexity of a de Bruijn graph $G = (V, E)$ is the sum of the finishing complexity of the repeat nodes:

$$C(G) = \sum_{v \in V} C(v)$$



Optical Maps

- DNA chromosome tagged with fluorescent marker is immobilized on a slide and washed with a restriction enzyme, which cuts the DNA at loci with a specific DNA sequence (4 – 8 bp). The cut locations are known as **restriction sites**.

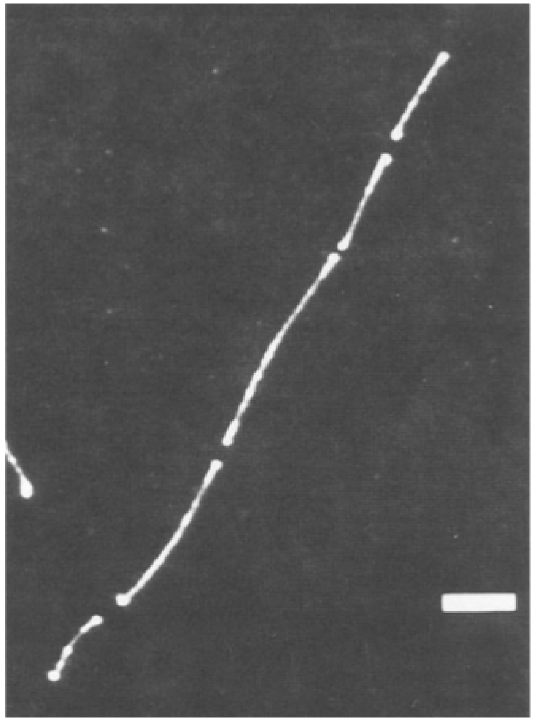
- The lengths of the resulting restriction fragments can be estimated by measuring the total fluorescence of each fragment.

- This experiment produces an **optical map**: an ordered listing of restriction fragment lengths along the chromosome.

- Experimental errors: false restriction sites or missing restriction sites.
  - Small restriction fragments will be missed

- The latest emerging optical mapping technology can provide limited sequence data near restriction sites.

- Optical map provides a constraint for allowable Eulerian tours.



Samad, A et al. (1995). Optical mapping: a novel, single-molecule approach to genomic analysis. Genome Research, 5(1), 1-4. doi:10.1101/gr.5.1.1

Algorithm Overview

1. Build a de Bruijn graph from DNA sequence

2. Simplify the de Bruijn graph using graph compression techniques

3. Extract a set of contigs from the de Bruijn graph

4. Align contigs to Optical Map

5. Use contig alignments to simplify graph

6. Evaluate reduction complexity in graph

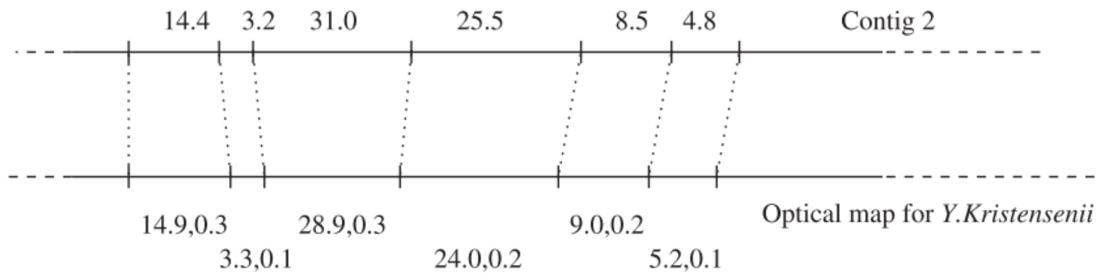
Contig to Optical Map Alignment

- Optical Map provides:
  - ordered list of restriction fragment lengths
  - standard deviations for each restriction fragment
  - 5 base pair reads around each restriction site

- Create **in-silico restriction sites** for a contig extracted from de Bruijn graph
  - Use contig DNA sequence and enzyme nucleotide recognition sequence

- Use dynamic programming to determine the best possible alignment of the contig to the optical map.

- Use a score function that scores an alignment based on:
  - Number of total missed restriction sites
  - Cumulative Levenshtein edit distance between 5 base pair reads around each aligned restriction site.
  - Chi-squared score for aligned fragment sizes



Nagarajan, N., Read, T. D., & Pop, M. (2008). Scaffolding and validation of bacterial genome assemblies using optical restriction maps. Bioinformatics (Oxford, England), 24(10), 1229-35. doi:10.1093/bioinformatics/btn102

Scoring Function

Each optical fragment length can be modeled as $\sim N(o_i, \sigma_i^2)$ where the experimentally measured length is o_i . The χ^2 scoring function for the alignment of insilico contig restriction fragments of lengths $c_1, \ldots, c_n$ to optical map fragments of lengths $o_j, \ldots, o_{j+n}$ with corresponding standard devations $\sigma_j, \ldots, \sigma_{j+n}$ is given by:

$$S_{\chi^2} = \sum_{i=1}^{n} \frac{(c_i - o_{j+i})^2}{\sigma_i^2}$$

The total match score is given by a sum of S_{χ^2}, the number of missed restriction sites m_r, the sum of the Levenshtein edit distance d_i, and constant weights C_d and C_r.

$$S = S_{\chi^2} + C_r \times m_r + C_d \times \sum_{i=1}^{n} d_i$$

The best match is given by the lowest score.

Dynamic Programming Algorithm

Let S_{ij} be the score of the best alignment matching the end of the ith contig fragment with the end of the jth optical map fragment. This score can be determined by considering extending previously scored alignments, and incorporating the appropriate edit distance and missed restriction site penalties for each possible extension:

$$S_{ij} = \max_{0 \le k \le i, 0 \le l \le j} C_r \times (i - k + l - j) + C_d \times d_j + \frac{\left(\sum_{s=k}^{i} c_s - \sum_{t=l}^{j} o_t\right)^2}{\sum_{t=l}^{j} \sigma_t^2} + S_{(k-1)(l-1)}$$

This produces an $\mathcal{O}(m^2n^2)$ algorithm, where m is the number of contig restriction fragments and n is the number of restriction fragments in the optical map. However, the search space can be pruned by not considering the extension of poor alignments.

Evaluating Alignments

• Can evaluate how significant an alignment is between a contig and the optical map through a **permutation test**

- Permute the restriction fragments of the contig and determine the best alignment score of the permuted contig

- Evaluate the probability that a permuted contig aligns better to the optical map than the original contig.

$$P\,(\text{alignment score of permuted contig} \geq \text{alignment score of original contig})$$

Algorithm Overview

1. Build a de Bruijn graph from DNA sequence

2. Simplify the de Bruijn graph using graph compression techniques

3. Extract a set of contigs from the de Bruijn graph

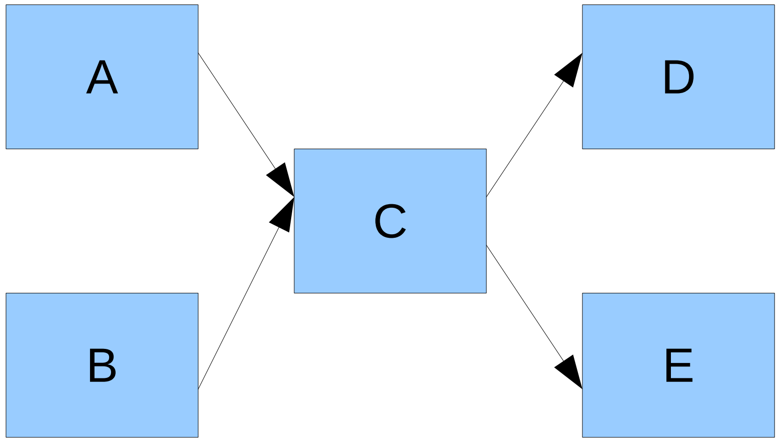
4. Align contigs to Optical Map

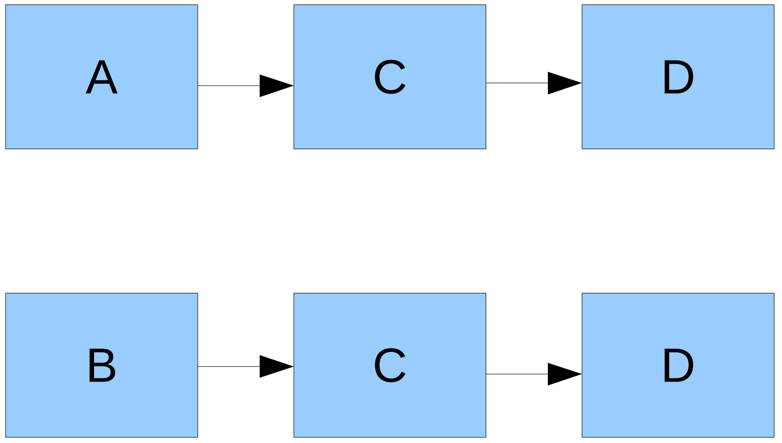
5. Use contig alignments to simplify graph

6. Evaluate reduction complexity in graph

Searching for the Path Between Contigs

- **Dijkstra's Algorithm** finds the shortest path between two vertices in a graph in O(E + V log V).
- The problem of finding a path of a given length between two vertices in a graph is NP-Hard.
- Simplify the de Bruijn graph if a shortest path between contigs is consistent with the optical map





Potential Issues

- Picking good values for the weighting constants that appear in the scoring function

- If contigs have multiple significant alignments to the optical map, will have to consider each possible alignment during graph simplification.

- It's possible that a path other than the shortest path might be the correct path between two contigs.

Implementation

- Code in C++
- Pipeline in Perl/Python
- Boost Graph Library (BGL) for graph implementation and graph search algorithms
- CBCB Computing Resources:
  - **Privet**: Sun Fire V40z (OEM Chassis)
  - 4 x AMD Opteron(tm) Processor 850 (2400MHz)
  - 32 GB Ram
  - RHEL5 x86_64

Databases

- NCBI Genbank (for reference genomes)
  - http://www.ncbi.nlm.nih.gov/genbank/
  - Can be used to generate an in-silico optical map of the reference genome

- Archive of simplified de Bruijn Graphs for prokaryotic genomes
  - http://www.cbcb.umd.edu/research/complexity/
  - (Kingsford, C., Schatz, M. C., & Pop, M. (2010). Assembly complexity of prokaryotic genomes using short reads. BMC bioinformatics, 11, 21. doi:10.1186/1471-2105-11-21)

- Experimental optical map data

Validation

• Can evaluate correctness of contig-optical map alignment and graph simplification techniques from user generated data

•Compare new contigs created through graph simplification to the known true DNA sequence.

Project Schedule & Milestones

•**<u>Phase I (Sept 5 – Nov 27)</u>**

- Complete code for the contig-optical map alignment tool
- Test algorithm by aligning user-generated contigs to user-generated optical map
- Begin implementation of Boost Graph Library (BGL) for working with assembly graphs.

<u>Phase II (Nov 27 – Feb 14)</u>

- Finish de Bruijn graph utility functions.
- Complete code for the assembly graph simplification tool
- Test assembly graph simplification tool on simple user-generated graph.

<u>Phase III (Feb 14 – April 1)</u>

- Verify performance of the contig-optical map alignment tool and the graph simplification tool with archive of de Bruijn graphs for reference bacterial genomes.
- Compute reduction in graph complexities.
- Verify performance using experimentally obtained optical maps

<u>Phase IV (time permitting)</u>

- Implement parallel implementation of the contig-optical map alignment tool using OpenMP
- Explore possibility of using the parallel Boost Graph Library.
- Test graph simplification tool on assembly graph produced by a de Bruijn graph assembler (ALLPATHS or SOAPdenovo).

Deliverables

- Source code for the contig alignment to optical map program
- Source code for the optical map simplification program
- Archive of simplified de Bruijn graphs for several bacterial genomes
- Final Report detailing software implementation, verification results, and reduction in complexity.

Sources

Kingsford, C., Schatz, M. C., & Pop, M. (2010). Assembly complexity of prokaryotic genomes using short reads. BMC bioinformatics, 11, 21.

Nagarajan, N., Read, T. D., & Pop, M. (2008). Scaffolding and validation of bacterial genome assemblies using optical restriction maps. Bioinformatics (Oxford, England), 24(10), 1229-35.

Pevzner, P. a, Tang, H., & Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences of the United States of America, 98(17), 9748-53.

Samad, a, Huff, E. F., Cai, W., & Schwartz, D. C. (1995). Optical mapping: a novel, single-molecule approach to genomic analysis. Genome Research, 5(1), 1-4.

Schatz, M. C., Delcher, A. L., & Salzberg, S. L. (2010). Assembly of large genomes using second-generation sequencing. Genome research, 20(9), 1165-73.

Wetzel, J., Kingsford, C., & Pop, M. (2011). Assessing the benefits of using mate-pairs to resolve repeats in de novo short-read prokaryotic assemblies. BMC bioinformatics, 12, 95.