Reducing Genome Assembly Complexity with Optical Maps Mid-year Progress Report

Lee Mendelowitz
LMendelo@math.umd.edu

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

December 18, 2011

Abstract

The goal of genome assembly is to reconstruct contiguous portions of a genome (known as contigs) given short reads of DNA sequence obtained in a sequencing experiment. De Bruijn graphs are constructed by finding overlaps of length $k-1$ between all substrings of length k from the reads, resulting in a graph where the correct reconstruction of the genome is given by one of the many possible Eulerian tours. The assembly problem is complicated by genomic repeats, which allow for exponentially many possible Eulerian tours, thereby increasing the de Bruijn graph complexity. Optical maps provide an ordered listing of restriction fragment sizes for a given 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 then use the constraints of these alignments to find paths through the assembly graph that resolve genomic repeats, thereby reducing the assembly graph complexity. The goal of this project is to implement the Contig-Optical Map Alignment Tool and the Assembly Graph Simplification Tool and to use these tools to simplify the idealized de Bruijn graphs for several bacterial genomes.

1 Introduction

Genome assembly is the computational task of determining the total DNA sequence of an organism given reads of DNA sequence obtained from a sequencing experiment. DNA is a double stranded helical molecule, where each strand comprises a sugar-phosphate backbone and a sequence of nucleobases. The four nucleobases are: Adenine (A), Thymine (T), Guanine (G), and Cytosine (C), with complementary base pairing between (Guanine, Cytosine) and (Adenine, Thymine). For each nucleobase along the primary strand, the complementary base appears in the same position of the complementary strand. Inside the cell, DNA molecules are condensed into a coiled secondary structure known as a chromosome. Due to the compact structure of the chromosome, the DNA bases are relatively inaccessible. As a result, experimentalists do not have

the ability to read an organism's DNA sequence directly from the chromosome itself. Instead, it is necessary to perform a DNA sequencing experiment.

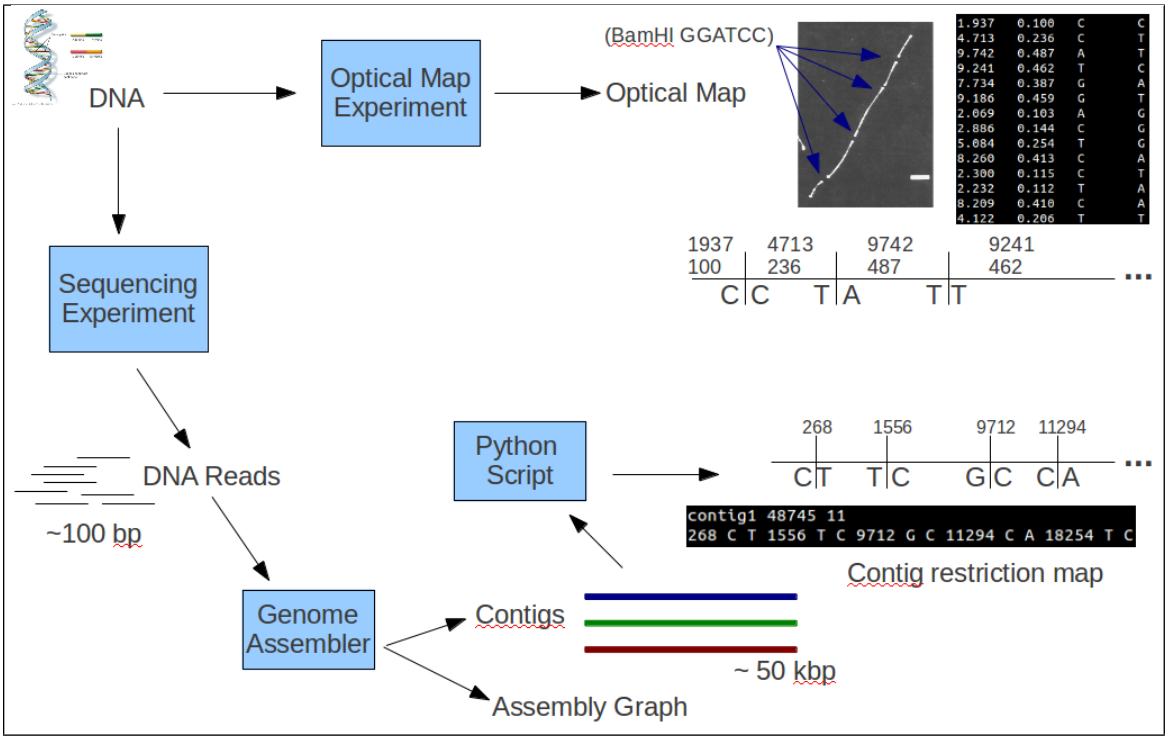

Figure 1: This figure provides the context for the algorithms to be developed in this project. A sequencing experiment produces short reads of DNA sequence from a set of chromosomes. Genome assembly software uses the reads to construct an assembly graph, and from the assembly graph outputs a set of contiguous DNA sequence known as contigs. In an optical mapping experiment, the same chromosomes are digested with a restriction enzyme, which cuts the DNA at sites with a specific recognition sequence, usually six base pairs in length. An optical map gives the restriction site pattern for each chromosome, providing the mean and standard deviation for each fragment length, as well as limited sequence information around each restriction site. An in-silico contig restriction map can be created for each assembled contig. The tools created as part of this project will align contigs to the optical map and search for unique paths through the assembly graph that are consistent with the optical map, thereby extending contig length and reducing the complexity of the assembly graph.

1.1 DNA Sequencing and Genome Assembly

In a DNA sequencing experiment, DNA is extracted from a sample and sheared into random fragments. The DNA sequence at the end of the fragments is determined experimentally, producing a set of reads. Today's sequencing technology is highly automated and parallelized, cheaply producing short reads of 35 to 400 base pairs long [6]. While the reads are error prone, sequencing machines deliver high throughput, providing many reads that cover the same portion of the genome. The task of genome assembly is often compared to the process of assembling a jigsaw

puzzle: given the DNA reads (i.e. the pieces), an assembler must assemble the unique genome (i.e. the picture) from which the reads originate. In practice, an assembler will not be able to perfectly reconstruct the entire genome, and instead will output a set of contiguous DNA sequences known as contigs.

Currently there are two different formulations of the genome assembly problem. In one formulation, known as overlap-layout-consensus (OLC), each read obtained in a sequencing experiment is compared to every other read in search of overlaps. This produces what is known as an overlap graph where reads are vertices and edges represent overlaps. A reconstruction of the genome is given as a Hamiltonian path through the graph that visits each node exactly once. Identifying Hamiltonian paths is NP-Hard, so algorithms that use OLC rely on various heuristics to produce contigs [4].

A second approach to genome assembly is to construct a de Bruijn graph from reads [3]. For a given value of $k > 1$, a de Bruijn graph is constructed by creating a vertex for each length k substring of DNA bases (known as a k-mer) that appears in a read. A directed edge is drawn from k-mer A to k-mer B if B follows A in a read (meaning k-mer B overlaps A by $k-1$ characters). Given perfect sequencing data such that each vertex in the graph is a true length k substring from the genome and each $k-1$ overlap is represented by exactly one edge, then a reconstruction of the genome is given by an Eulerian tour through the graph that uses each edge exactly once. From the construction of the de Bruijn graph, it should be clear that the in-degree of a vertex is equal to the out-degree, except for the starting and ending vertex corresponding to the beginning and end of a chromosome. For the case of a circular chromosome, each vertex will have equal in- and out-degrees.

In contrast to Hamiltonian paths, Eulerian tours are easy to identify and can be found in linear time [3]. The de Bruijn graph formulation of genome assembly has the advantage that it avoids the computationally expensive pairwise comparison of all reads. However, it comes with the disadvantage that information of the sequence of each read is lost after k-mers are formed. In addition, a de Bruijn graph structure is highly sensitive to sequencing errors, which can create false k-mers and therefore extraneous nodes and edges in the graph.

De Bruijn graphs can be simplified from the original graph of k-mer vertices through a series of graph compression techniques which reduce the number of nodes and edges but leave the DNA sequence spelled by each possible Eulerian tour unchanged (see [1]). Even with graph compression techniques, the number of possible Eulerian tours for a given graph is exponential in the number of repeat vertices (i.e. vertices which must be traversed multiple times in an Eulerian tour). The number of potential genome reconstructions ending at vertex t of a de Bruijn Graph $G = (V, E)$ with vertex set V, and collection of directed edges E can be determined exactly [1]. Let $\boldsymbol{A}$ be the adjacency matrix of G, and let $d^-(u)$ and $d^+(u)$ be the in-degree and out-degree of vertex u. Define $\boldsymbol{r}$ such that $r_t = d^+(t) + 1$ for vertex t and $r_u = d^+(u)$ for all vertices $u \neq t$. The Laplacian matrix of G is given by $\boldsymbol{L} = \texttt{diag}(\boldsymbol{r}) - \boldsymbol{A}$. Then the number of possible linear genome reconstructions W corresponding to Eulerian tours through G ending at vertex t is given by:

$$W(G,t) = \det L \left\{ \prod_{v \in V} (r_u - 1)! \right\} \left\{ \prod_{(u,v) \in E} a_{uv}! \right\}^{-1} \tag{1}$$

For the case of a circular chromosome, the number of genome reconstructions is given by $W(G,t)/d^+(t)$ since each circular genome has $d^+(t)$ linear representations that end at vertex t.

As an example of the application of (1), consider the simplified de Bruijn graph with $k = 100$ for the single circular chromosome of the bacterium *Mycoplasma genitalium*, which has the smallest known cellular genome at 580,076 base pairs. The de Bruijn graph has 84 vertices and 120 edges where 28 vertices are repeated twice and 4 vertices are repeated three times in each Eulerian tour. The number of unique genome reconstructions from this de Bruijn graph for the smallest known cellular genome is a staggering 21,897,216.

Since the number of possible genome reconstructions for the typical de Bruijn graph is so large, it is useful to define a different metric that measures the repeat structure of the graph. For a given repeat vertex v with $d^+(v) = d^-(v) = a$, the 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 [7]:

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

The sum is from $i = 2$ to $i = a$ since once these incoming edges are matched to an outgoing edge, the last remaining incoming edge is automatically matched to the last remaining outgoing edge. The total finishing complexity of a de Bruijn graph $G = (V, E)$ is the sum of the finishing complexity of the repeat nodes:

$$C(G) = \sum_{v \in V} C(v) \tag{3}$$

Since the number of Eulerian paths is exponential in the number of repeat vertices, the genome assembly task is to identify the one Eulerian tour of many possible tours that gives the correct reconstruction of the genome. Additional experimentally obtained information must be used to provide constraints on the allowable tours. Experimentally obtained information frequently used for this purpose are pairs of reads, known as mate pairs, that are separated by an approximately known distance in the genome [4]. Instead, in this project, we consider using information provided by an optical map.

1.2 Optical Mapping Technology

An optical map is produced by immobilizing a DNA molecule tagged with fluorescent marker on a slide [5]. A restriction enzyme is washed over the slide, and the enzyme cuts the DNA at loci with a particular 4 to 8 base pair recognition sequence known as a restriction site, producing a set of restriction fragments whose lengths are measured. As an example, the restriction enzyme PvuII has recognition sequence `CAGCTG`, so PvuII will cut the DNA molecule wherever this sequence of bases occurs. An optical map provides an ordered list of restriction fragment sizes along the length of the DNA molecule. Current state of the art optical mapping techniques can also provide limited DNA sequence data around the restriction sites. Experimental errors from the optical mapping process include missing restriction sites due to partial enzyme digestion of the molecule, false restriction sites that do not correspond to the true recognition sequence, and missing small restriction fragments. Despite these errors, optical maps provide useful long range information over the entire length of a DNA molecule - information that can be used to determine segments of the correct Eulerian tour through the de Bruijn graph.

2 Project Objectives

The goal of this project is to build software that uses optical maps with limited sequence information around each restriction site to resolve repeats in an idealized de Bruijn assembly graph, thereby extending the length of contigs and reducing the number of edges and nodes in the graph. The first goal is to get a working release of the Contig-Optical Map Alignment Tool and the Graph Simplification Tool by the conclusion of Phase II. The algorithms will be individually validated on user generated data sets. In Phase III, a pipeline will be created to use these tools together to simplify the idealized de Bruijn graphs for known bacterial reference genomes using simulated optical maps. In Phase IV of the project, a parallel implementation of the Contig-Optical Map Alignment Tool will be implemented, as time permits.

3 Approach

The software produced for this project consists of two major components. The first component finds alignments of contigs extracted from the de Bruijn graph to the optical map. The second component will use contig alignments to simplify the de Bruijn graph.

3.1 Contig-Optical Map Alignment Tool

The Contig-Optical Map Alignment Tool finds significant alignments of contigs extracted from a de Bruijn graph to a single optical map. Contigs can easily be extracted from the de Bruijn graph by reading the sequence along unambiguous paths that do not pass through any repeat vertex. Working with de Bruijn graphs that have been simplified through the graph compression techniques from [1], contigs are given by the concatenated DNA sequence of neighboring vertices. The inputs to this tool will be the DNA sequence for each contig to be aligned, the optical map data, and the recognition sequence of the enzyme used to produce the optical map. The optical map data consists of an ordered list of restriction fragments lengths, the standard deviations of the restriction fragment lengths, and several bases of sequence data on each side of the restriction site.

First, "in-silico" restriction sites are identified along each contig using the known recognition sequence of the enzyme used to produce the optical map. This produces an ordered list of restriction fragment lengths that would be produced if the contig were to be perfectly digested by the same enzyme used to produce the optical map. Next, each contig is aligned to the optical map through a dynamic programming algorithm which scores alignments based on a comparison of aligned restriction fragment lengths, the number of missed restriction sites, and the Levenshtein edit distance between the DNA sequence around the aligned restriction sites of the contig and optical map. Lastly, the statistical significance of each alignment is evaluated through a permutation test.

The total alignment score is given by a weighted sum of the χ^2 score S_{χ^2}, the number of missed restriction sites m_r, and the sum of the Levenshtein edit distance at the aligned restriction sites S_d, with constant weights C_s and C_r. The best match is given by the lowest score.

$$S = S_{\chi^2} + C_r \times m_r + C_s \times S_d \tag{4}$$

The χ^2 score S_{χ^2} measures the sum of the squared difference of restriction fragment lengths, where the length differences are measured in standard deviation units. Each true optical fragment length can be modeled as $\sim N(o_i, \sigma_i^2)$ where o_i is the mean and σ_i^2 is the variance of the fragment length obtained from repeated experimental measurements. The χ^2 scoring function for the alignment of in-silico contig restriction fragments of lengths $c_0, \ldots, c_{n-1}$ to optical map fragments of lengths $o_j, \ldots, o_{j+n}$ with corresponding standard deviations $\sigma_j, \ldots, \sigma_{j+n}$ is given by:

$$S_{\chi^2} = \sum_{i=0}^{n-1} \frac{(c_i - o_{j+i})^2}{\sigma_{j+i}^2} \tag{5}$$

The Levenshtein edit distance is the minimum distance between two character strings using a "+1" cost for each character deletion, insertion, or substitution, and is computed using dynamic programming in $\mathcal{O}(mn)$ for comparison of two stings with lengths m and n. S_d represents the total edit distance at aligned sites:

$$S_d = \sum_{i=0}^{n-2} d_{i,j+i} \tag{6}$$

where $d_{i,j+i}$ is the edit distance of the ith aligned site. Note that the sum in (6) only includes a total of $n-1$ terms since when n fragments are aligned, $n-1$ restriction sites are aligned.

3.1.1 Dynamic Programming

The scoring function given by (4) admits a dynamic programming algorithm to find the best possible alignment of the entire contig to the optical map through the jth restriction fragment of the optical map [2]. 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:

$$S_{ij} = \min_{0 \le k \le i, 0 \le l \le j} C_r \times (i - k + j - l) + C_s \times d_{ij} + \frac{(\sum_{s=k}^{i} c_s - \sum_{t=l}^{j} o_t)^2}{\sum_{t=l}^{j} \sigma_t^2} + S_{(k-1)(l-1)} \tag{7}$$

where d_{ij} is the edit distance for the alignment of the ith restriction site from the contig to the jth restriction site from the optical map.

For a given k and l, the score given by (7) is for an alignment where optical map restriction fragments k thru i are considered to be one restriction fragment with $k-i$ false-positive restriction sites, and contig restriction fragments l thru j are considered to be one contig restriction fragment with $j-l$ restriction sites missing from the optical map. This gives 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. An example of this process is illustrated in Figure 2 below.

3.1.2 Additional Alignment Constraints

The search space is pruned by not scoring alignments where an aligned fragment has a poor χ^2 score in (7):

$$\frac{(\sum_{s=k}^{i} c_s - \sum_{t=l}^{j} o_t)^2}{\sum_{t=l}^{j} \sigma_t^2} \ge C_\sigma \tag{8}$$

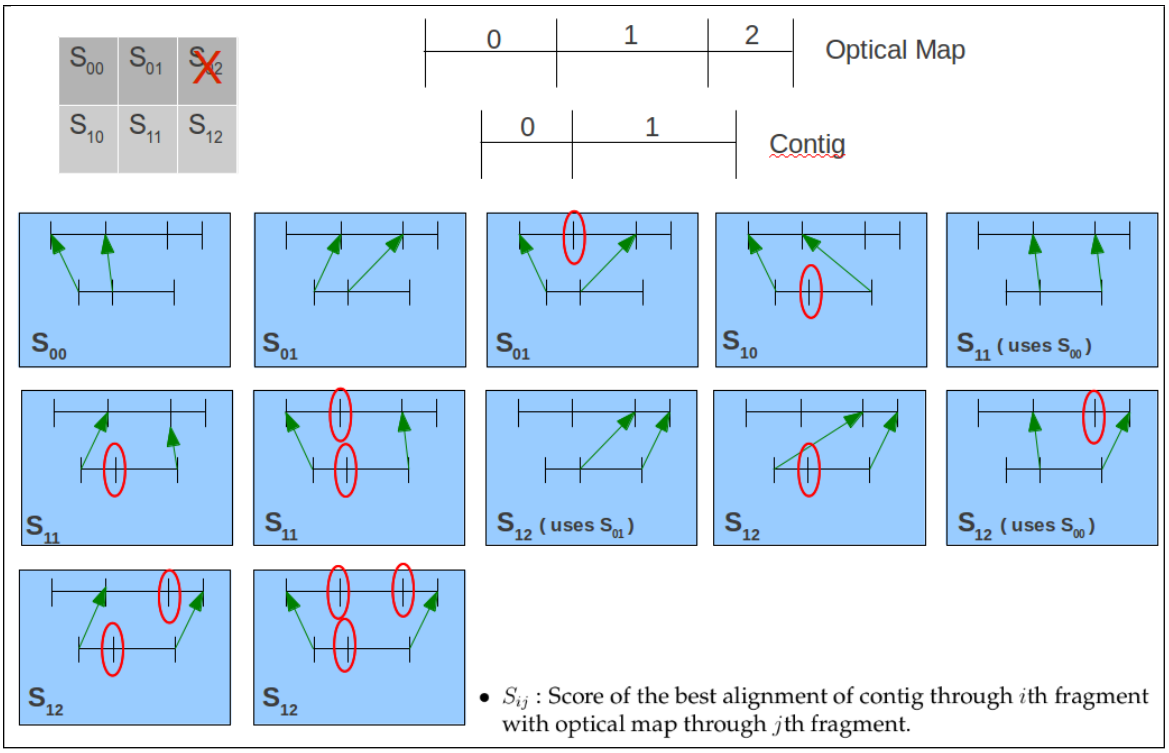

Figure 2: The different alignments that are scored in setting each element S_{ij} using equation (7) for two contig fragments ($m = 2$) and three optical map fragments ($n = 3$). Green arrows indicate the alignment of sites. Restriction sites circled in red count as a missed site for the alignment being scored. S_{02} represents the score of the alignment of the zeroth contig fragment to the second optical map fragment. This alignment is not considered because this means the first fragment of the contig would not be aligned to the optical map.

C_σ is a user input, which defaults to 5.

Since DNA is double stranded, it is possible that the optical map represents the locations of restriction sites along one strand while the contig represents the sequence from the complementary strand. To test for this possibility, both the contig and the reverse-complement of the contig are aligned to the optical map and the best possible alignment is selected. The reverse-complement of the contig restriction pattern is found by reversing the order of the fragments and taking the reverse complement of the sequence around each restriction site. In addition, to capture the periodic nature of a circular chromosome, the optical map must be doubled in length where each of the original n fragments is duplicated after the nth fragment. This allows for alignments that extend beyond the right edge of the original linearized optical map, which is a valid alignment for a circular genome.

To test the significance of the best alignment, a permutation test is performed by computing the best alignment score for the contig with its restriction fragments permuted. 500 samples from the space of permuted contigs is used to calculate the p-value. The fraction of permuted contigs that score better than the original contig gives the p-value, and an alignment is deemed significant if $p < 0.05$. Note that the permutation test is an expensive operation, as it increases run time by a

factor of 500.

3.2 Assembly Graph Simplification Tool

The Assembly Graph Simplification Tool will take as input the de Bruijn graph, a list of significant contig-optical map alignments as found by the Contig-Optical Map Alignment Tool, the optical map data, and the recognition sequence of the enzyme used to produce the optical map. From these inputs, the tool will output a simplified de Bruijn graph. For each pair of aligned contigs i and j, the Assembly Graph Simplification Tool will use Dijkstra's algorithm to find the shortest path through the de Bruijn graph from the vertex corresponding to contig i to the vertex corresponding to contig j. We rely on this heuristic of the shortest path since, in general, finding a path of a specified length through a graph is NP Hard [7]. The significance of this path can be evaluated by forming the contiguous DNA sequence along the path, and computing the sequence's alignment score to the optical map using (4) and the p-value of the alignment.

After calculating the shortest path and p-value for each pair of aligned contigs, the paths that have the best p-values will be greedily accepted as true. The de Bruijn graph will be updated by merging the vertices and edges along the path from contig i to contig j into a single vertex v_{new}. As a result, the complexity (given by (2)) of any repeat vertex v along this path will be reduced since at least one incoming and outgoing edge to/from v will be removed in the creation of v_{new}.

One predictable issue that might be encountered when using the Assembly Graph Simplification Tool is if the shortest path between contigs is not the correct path. In this case, the shortest path between the contigs should not produce a significant alignment to the optical map, and so no simplifications will be made to the de Bruijn graph for this contig pair. Another predictable issue is if the Contig-Optical Map Alignment Tool produces many potential alignments for a single contig. In this case, the Assembly Graph Simplification Tool may have to consider only best alignment to save computation time.

4 Implementation

Both the Contig-Optical Map Alignment Tool and the Assembly Graph Simplification Tool will be programmed in C++.

4.1 Contig-Optical Map Alignment Tool

The Contig-Optical Map Alignment Tool has been developed by updating source code from the open-source software package SOMA (described in [2]), which is a tool for matching and placing multiple contigs along an optical map. Code development was performed using the Vim text editor in a Linux environment. Development was performed on my home desktop computer (Ubuntu 11.10), on my MacBook Pro (Ubuntu 11.10 in VirtualBox), and on the Center for Bioinformatics and Computational Biology's computational resources. In mid-November, in an effort to keep my code in sync across these platforms, I created a local Git repository which is now hosted privately for free with `BitBucket.org`. Git is a distributed version control system that keeps a history of file changes and provides branching and merging functionality. The code was compiled using the `g++` compiler.

Development of the Contig-Optical Map Alignment tool involved several important tasks. The first task was to understand the existing alignment algorithm as implemented in the SOMA source code. While the source code itself lacked documentation, the reference [2] outlined the algorithm.

The second task was to reorganize and restructure the code to improve modularity and maintainability without changing the functionality of the original SOMA program. In particular this involved organizing global variables and constant variables into a separate header. C structs were converted into C++ classes to provide more flexibility. I also made use of the C++ Standard Template Library (STL) wherever possible in an effort to make the code easier to maintain. With each set of changes, I would recompile the software using `g++` and re-run the test case provided with the SOMA distribution to ensure that the results were unchanged.

The third task was to make functionality updates to the Contig-Optical Map Alignment Tool. This included writing functions to calculate the Levenshtein edit distance between strings and to compute the reverse complement of a DNA string. The alignment scoring function was updated to include the edit distance of sequence around each restriction site. In addition, functions to read input files and write output files were modified to handle sequence data.

During testing of the tool, it was discovered that execution time with the permutation test was less than desirable. The GNU profiling tool `gprof` was used to determine which function calls took most of the execution time. It was determined that a staggering number of calls were made to the Levenshtein edit distance function, consuming approximately 50% of the execution time. The computation of the alignment was reformulated to calculate the edit distance of the sequence flanking an aligned restriction site as opposed to the edit distance of the sequence at the beginning and end of an aligned restriction fragment. This simple change allowed the calculation of edit distance to be made independently of the calculation specified by (7) (since d_{ij} does not depend on l or k). This resulted in significantly less calls to the edit distance function. During this phase of the code optimization it was also discovered that opportunities existed to continue to the next iteration of a loop or break out of a loop when the condition specified by (8) was not met. Lastly, in the case of a circular chromosome, updates were made to ensure that no alignment is started beyond the right edge of the linear representation of the circular map. These code changes save unnecessary computation, yielding a more efficient program.

Lastly, updates were made to SOMA's original pre-processing pipeline. The script for generating the in-silico restriction sites for each contig was converted from perl to python for easier maintainability.

The software was implemented and tested using the Center for Bioinformatics and Computational Biology's computing resource `privet`: 4 x AMD Opteron(tm) Processor 850 (2400MHz), 32 GB Ram, RHEL5 x86_64.

4.2 Graph Simplification Tool

The Assembly Graph Simplification Tool will use the Boost Graph Library, which provides useful classes for constructing graphs as well as graph search algorithms. The Boost Graph Library is open source and is available at `http://www.boost.org`. Implementation of this tool is currently underway, to be completed at the conclusion of Phase II.

5 Databases

Source code, scripts, and validation results are being maintained in a private git repository, hosted at `BitBucket.org`.

Reference genomes for bacterial genomes are available from the National Center for Biotechnology Information's (NCBI) Genbank: `http://www.ncbi.nlm.nih.gov/genbank/`. These reference genomes can be used to construct artificial optical maps to test the tools developed as part of this project.

The input de Bruijn Graphs to be used for testing will be the simplified de Bruijn graphs found in [1], available at: `http://www.cbcb.umd.edu/research/complexity/`

6 Validation

6.1 Contig-Optical Map Alignment Tool

The Contig-Optical Map Alignment tool was tested on two user generated data sets to validate its performance.

6.1.1 Test 1

An optical map of 100 fragments was generated using a python script. The fragment lengths were selected at random uniformly from $[1.0, 10.0]$ kbp, the standard deviations were set to 1 bp (to simulate as little sizing error as possible), and sequence data of 1 bp preceding and following each restriction site was generated uniformly at random from the alphabet {A,C,G,T}. 10 contigs were extracted from this map, with probability of 0.5 in the forward orientation and probability 0.5 in the reverse orientation. The extraction locations were selected uniformly at random from the optical map, and the number of fragments per contig was selected uniformly at random from $[5, 20]$. In addition, 10 contigs were randomly generated using the same parameters described above. The permutation test was turned off. The Contig-Optical Map Alignment tool was run with $C_\sigma = 5$, $C_r = C_s = 12500$ to align the 10 true contigs and 10 random contigs to the generated optical map. These weights effectively ignore the χ^2 score component of the scoring function since the sequence error and restriction site error terms are weighted so highly. The algorithm correctly aligned the 10 true contigs to the correct location in the optical map, and the best scoring alignments for the random contigs were of poor quality, with many missed restriction sites. This test indicates that the Contig-Optical Map Alignment tool works correctly on perfect data.

6.2 Test 2

An optical map of 400 fragments was generated using a python script. The fragment lengths were selected at random uniformly from $[1.0, 10.0]$ kbp, the standard deviations were selected uniformly at random to up to 5% of the mean fragment length, with a minimum standard deviation of 100 bp and a maximum of 500 bp. 30 contigs were extracted from the map, as in Test 1, but with simulated errors described as follows. Sizing error was added to each contig fragment length, sampled uniformly from the range specified by $\pm 5\%$ of the mean fragment length given in the optical map. A substitution error was made for each base in the sequence data with probability

0.1. A restriction site error is made with probability 0.1 by deleting a site in the contig or by inserting a false site in the contig halfway between consecutive sites. 10 contigs were also generated randomly as described previously. The permutation test was turned on, with 500 samples taken from the space of permuted contigs using a p-value threshold of 0.05.

The Contig-Optical Map Alignment tool was run with $C_\sigma = 5$, $C_r = C_s = 12500$. The 30 true contigs aligned to the correct position in the map. 1 out of 10 of the random contigs aligned to the map with significance (a false positive), with a p-value of 0.03. Of the 9 restriction sites in this random contig, 3 were not aligned to restriction sites in the optical map. With only 6 aligned restriction sites, the random contig had, by chance, a very good sequence score, which is why it performed well on the permutation test. The χ^2 score for the alignment was very poor, 67.7 for only 5 inner fragments.

This test was repeated using the same input optical map and set of contigs, with updated costs $C_r = 5$ and $C_s = 3$, which gives more importance to the χ^2 score. With these updated costs, the best alignment for the previous false positive was no longer deemed significant. However, these updated costs introduced a new false positive, with p-value 0.046. This new false positive contig has 9 sites, 3 of which were not aligned to the optical map.

I believe that these false positives can be eliminating by putting a constraint on the fraction of restriction sites that are allowed to be missed (say 10%). This constraint can be used to prune the search space (just as the condition on fragment length specified by (8)). This constraint on missed restriction sites will also improve the run time of the alignment algorithm. More testing will be necessary to determine how effective this constraint is at eliminating false positives. I have added a milestone to Phase II of the project to investigate this further.

6.3 Assembly Graph Simplification Tool

The Assembly Graph Simplification Tool can be validated by testing it on the de Bruijn graph generated for the a truncated reference genome. Since we know the true sequence between any two contigs, we can evaluate each resulting simplification of the de Bruijn graph for correctness.

7 Testing on new Databases

If time permits, the tools developed as part of this project can be tested on de Bruijn graphs generated by a real assembler (SOAPdenovo or ALLPATHS) on simulated reads with sequencing error from a known reference genome.

8 Project Schedule & Milestones

Phase I of the project is now complete. Version 1.0 of the Contig-Optical Map Alignment Tool has been completed. The Boost Graph Library has been downloaded and compiled. I have added a milestone for Phase II to test the effects of adding a maximum missed restriction site constraint to the Contig-Optical Map Alignment Tool, as discussed in section 6.2. This new functionality will be released with version 1.1 of the Contig-Optical Map Alignment Tool.

- Phase I (Sept 5 to Nov 27)

  - Complete code for the Contig-Optical Map Alignment Tool (Completed version 1.0)
  - Validate performance of algorithm by aligning user-generated contigs to a user-generated optical map (Completed for version 1.0)
  - Begin implementation of Boost Graph Library (BGL) for working with assembly graphs. (Completed)

- Phase II (Nov 27 to Feb 14)
  - Test effects of maximum missed restriction site constraint on false positive alignments. Complete version 1.1 of Contig-Optical Map Alignment Tool.
  - Finish de Bruijn graph utility functions.
  - Complete code for the Assembly Graph Simplification Tool
  - Validate performance of Assembly Graph Simplification Tool on de Bruijn graph from known reference genome

- Phase III (Feb 14 to April 1)
  - Create pipeline for using Contig-Optical Map Alignment Tool and Assembly-Graph Simplification Tool together on one dataset.
  - Test performance of the Contig-Optical Map Alignment Tool and Assembly Graph Simplification Tool with archive of de Bruijn graphs for reference bacterial genomes and artificial optical maps.
  - Compute reduction in graph complexities for each reference bacterial genome
  - Verify performance using experimentally obtained optical maps

- Phase IV (time permitting)
  - Implement parallel implementation of the Contig-Optical Map Alignment Tool using OpenMP
  - Explore possibility of using the parallel Boost Graph Library.
  - Test Assembly Graph Simplification Tool on assembly graph produced by a de Bruijn graph assembler (ALLPATHS or SOAPdenovo).

9 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 and validation results.

References

[1] Carl Kingsford, Michael C Schatz, and Mihai Pop, *Assembly complexity of prokaryotic genomes using short reads.*, BMC bioinformatics **11** (2010), 21.

[2] Niranjan Nagarajan, Timothy D Read, and Mihai Pop, *Scaffolding and validation of bacterial genome assemblies using optical restriction maps.*, Bioinformatics (Oxford, England) **24** (2008), no. 10, 1229–35.

[3] P A Pevzner, H Tang, and M S Waterman, *An Eulerian path approach to DNA fragment assembly.*, Proceedings of the National Academy of Sciences of the United States of America **98** (2001), no. 17, 9748–53.

[4] Mihai Pop, *Genome assembly reborn: recent computational challenges.*, Briefings in bioinformatics **10** (2009), no. 4, 354–66.

[5] A. Samad, E. F. Huff, W. Cai, and D. C. Schwartz, *Optical mapping: a novel, single-molecule approach to genomic analysis.*, Genome Research **5** (1995), no. 1, 1–4.

[6] Michael C Schatz, Arthur L Delcher, and Steven L Salzberg, *Assembly of large genomes using second-generation sequencing.*, Genome research **20** (2010), no. 9, 1165–73.

[7] Joshua Wetzel, Carl Kingsford, and Mihai Pop, *Assessing the benefits of using mate-pairs to resolve repeats in de novo short-read prokaryotic assemblies.*, BMC bioinformatics **12** (2011), 95.