Reducing Genome Assembly Complexity with Optical Maps

Lee Mendelowitz
LMendelo@math.umd.edu

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

October 11 2011

Abstract

De Bruijn graphs provide a framework for genome assembly, where the correct reconstruction of the genome is given by one of the many Eulerian tours through the graph. The assembly problem is complicated by genomic repeats, which allow for 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.

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.

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.

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.

Approach

The software produced for this project will consist of two major components. The first component will find 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.

Contig-Optical Map Alignment Tool

The Contig-Optical Map Alignment Tool will find significant alignments of contigs extracted from a de Bruijn graph to a single optical map. Contigs (i.e. contiguous DNA sequence) 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 five base pairs of sequence data on each side of the restriction site.

First, "in-silico" restriction sites will be identified along each contig using the known recognition sequence of the enzyme used to produce the optical map. This will produce 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 will be aligned to the optical map through a dynamic programming algorithm (described below) which scores alignments based on a comparison of restriction fragment lengths, the number of restriction fragments, 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 will be evaluated through a permutation test.

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 $c_1, \ldots, c_n$ 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:

$$S_{\chi^2} = \sum_{i=1}^{n} \frac{(c_i - o_{j+i})^2}{\sigma_i^2} \tag{4}$$

The χ^2 scoring function measures the sum of the squared difference of restriction fragment lengths, where the length differences are measured in standard deviation units.

The total alignment score is given by a weighted sum of S_{χ^2}, the number of missed restriction sites m_r, and the sum of the Levenshtein edit distance d_i, with constant weights C_d and C_r. The Levenshtein edit distance is the minimum distance between two character strings using a "+1" cost for each character deletion, insertion, or substitution, and can be computed using dynamic programming in $\mathcal{O}(mn)$ for comparison of two stings with lengths m and n.

$$S = S_{\chi^2} + C_r \times m_r + C_d \times \sum_{i=1}^{n} d_i \tag{5}$$

The best match is given by the lowest score.

The scoring function given by (5) admits a dynamic programming algorithm to find the best possible alignment of the entire contig to the optical map up to the jth restriction fragment [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.

$$S_{ij} = \max_{0 \le k \le i, 0 \le l \le j} C_r \times (i - k + j - l) + C_d \times d_j + \frac{\left(\sum_{s=k}^{i} c_s - \sum_{t=l}^{j} o_t\right)^2}{\sum_{t=l}^{j} \sigma_t^2} + S_{(k-1)(l-1)} \qquad (6)$$

For a given k and l, the score given by (6) 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 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 [2].

To test the significance of each possible contig-optical map alignment, a permutation test can be performed by computing the best alignment score for the contig with its restriction fragments permuted. The probability that a permuted contig scores better than the original contig can be found by sampling from the space of permuted contigs. The p-value is given by: $p = P(\text{alignment score of permuted contig} \geq \text{alignment score of original contig})$, and an alignment is deemed significant if $p < 0.05$.

One predictable issue that might be encountered when using the Contig-Optical Map Alignment Tool is selecting good values of weighting constants C_r and C_d in the scoring function (5).

The Contig-Optical Map Alignment Tool will be 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.

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 (5) 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.

Implementation

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

Pre-processing and post-processing scripts will be written in Perl and/or Python.

The software will be 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.

Databases

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/`

Validation

The Contig-Optical Map Alignment Tool can be validated by testing it on user-generated data where the correct alignments are known. An artificial optical map can easily be created from a truncated reference genome, and random contigs can be extracted from the known reference genome to be aligned to the map. Since the true alignment of the contig is known, the tool's alignments can be evaluated for correctness.

Similarly, the Assembly Graph Simplification Tool can be validated by testing it on the de Bruijn graph generated for the same 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.

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.

Project Schedule & Milestones

- Phase I (Sept 5 to Nov 27)
  - Complete code for the Contig-Optical Map Alignment Tool
  - Validate performance of algorithm by aligning user-generated contigs from a short reference genome to user-generated optical map
  - Begin implementation of Boost Graph Library (BGL) for working with assembly graphs.
- Phase II (Nov 27 to Feb 14)
  - 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)
  - 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).

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.