A De Bruijn Representation of a Repeat Family: Project Proposal

Sindhuja Parimalarangan, Matthew Allan, Scott Cheloha

October 12, 2015

1 Introduction

De Bruijn graphs are used in computational biology to assemble collections of short reads into genome sequences. This approach is confounded by repetitive elements, which create cycles in the graph. However, we believe we can actually take advantage of the properties of De Bruijn graphs in order to represent a set of related repeat elements and to derive a consensus (i.e. average) form of the repeat element.

For our project, we will implement a De Bruijn graph representation of a repeat family.

2 Motivation

While other representations of repeat families exist, there is plenty of room for experimentation toward a better, perhaps standard, representation.

For this project, as mentioned above, we will use De Bruijn graphs as a basis for developing such a representation. De Bruijn graphs are presently used

to assemble short reads. We want to explore whether it is possible to alter and/or augment the De Bruijn structure to better represent repeat elements.

For instance, existing consensus sequences give the probability of finding a base at a given position, but do not give the probability of finding a given base in the position immediately *after* said base. De Bruijn graphs could show, e.g., that an Adenine at position 1 is almost always followed by an Adenine at position 2, even if the probabilities of finding an Adenine at positions 1 or 2 is only 10% for either position.

3 Goals

In order of decreasing priority, we want to:

1. Construct a De Bruijn graph representation of a chosen repeat family;
2. Develop a representation for a set of related repeat elements and derive a consensus (i.e. average) form for the repeat element; and
3. Design and implement an optimized algorithm for representing an input repeat family using De Bruijn graphs.

4 Resources

The following subsections list the resources we anticipate will be necessary to achieve the prenominated goals. The lists will grow with our understanding of the problem.

4.1 Data

At a minimum we will use the following datasets:

- GenBank
- LINE/SINE element sequences of indeterminate source

4.2 Hardware

We will probably need time on a cluster of some capacity in order to execute our software on larger datasets.

4.3 Software

Exactly what form the output of this project will take with regard to implementation language, use of preexisting libraries, if any, etc., is unclear. For now, at least while researching for the project and preparing the final report, we believe we will use the following:

- UCSC Genome Browser, for lots of things;
- Galaxy, for data analysis;
- RepeatMasker, as a reference implementation of some aspects of the project;
- Velvet and related software, for sequence assembly;
- RepeatFinder, RECON, and RepeatGluer, for finding and processing repeats;
- Some sort of graph visualization library, for algorithm design and the final report; and
- The GIMP, for manipulating those visualizations.

5 Responsibilities

Each team member will be partly responsible for determining how to use De Bruijn graphs to represent a consensus sequence for our target repeat families. Each team member will also be partly responsible for deciding upon which axes (computational efficiency, output accuracy, quality of output representation, etc.) we will compare our results to those of RepeatMasker.

In addition, each team member will be responsible for a more specific partition of tasks tuned to their interests and abilities.

Sindhuga Parimalarangan will:

- Design and implement a software workflow for collecting, parsing, and storing data;
- Optimize the algorithm vis a vis data structures and external libraries to mitigate redundant searching; and
- Implement both the graph construction and consensus sequence-finding algorithms.

Matthew Allan will:

- Identify repeat target repeat families and document why they are suitable for the project;
- Determine what RepeatMasker *does* in order to determine a consensus sequence so that we something of a reference;
- Run RepeatMasker as needed; and
- Explain any differences between RepeatMasker's output and that of our implementation.

Scott Cheloha will:

- Scour the literature for De Bruijn graph construction algorithms;
- Determine which of those algorithms exist in a form we can use (legally, architecturally, etc.) and which we must implement ourselves; and
- Implement both the graph construction and consensus sequence-finding algorithms.

6 Timeline

On **November 9, 2015**, we will have:

1. Identified our target family/families of repeats;
2. Generated De Bruijn graphs for the same using some preexisting implementation;
3. Obtained any needed datasets and stored them (possibly on the cluster) in a suitable format;
4. Run RepeatMasker and learned what we can from it; and
5. Finalized the high-level details of the algorithm we intend to implement.

On **December 12, 2015**, we will have:

1. A clean, optimized implementation of our algorithm;
2. The results of our algorithm in a legible format; and
3. Determined how (or how *not*) to make a consensus sequence from De Bruijn graphs that is more informative than a table of letter frequencies.