Genome Polymorphism Detection Through Relaxed de Bruijn Graph Construction

M. Stanley Fujimoto*, Cole Lyman*, Anton Suvorov†, Paul Bodily*,
Quinn Snell*, Keith Crandall‡, Seth Bybee† and Mark Clement*
*Computer Science Department
Brigham Young University,
Provo, UT USA
Email: sfujimoto@gmail.com
†Biology Department
Brigham Young University,
Provo, UT USA
‡Computational Biology Institute
George Washington University,
Washington, D.C. USA

***Abstract*—Comparing genomes to identify polymorphisms is a difficult task, especially beyond single nucleotide polymorphisms. Polymorphism detection is important in disease association studies as well as in phylogenetic tree reconstruction. We present a method for identifying polymorphisms in genomes by using a modified version de Bruijn graphs, data structures widely used in genome assembly from Next-Generation Sequencing. Using our method, we are able to identify polymorphisms that exist within a genome as well as well as see graph structures that form in the de Bruijn graph for particular types of polymorphisms (translocations, etc.)**

***Keywords*-de Bruijn graph, graph, GWAS, phylogenetics, polymorphism**

I. Introduction

Detecting polymorphisms in the genome is an important task for an individual specimen (disease association studies) and for a species as whole (phylogenetic tree reconstruction). Whether it be identifying single nucleotide polymorphisms (SNPs) in an individual compared to a reference genome or comparing different species, identifying polymorphic differences is a difficult task. Methods, however, are usually extremely conservative and only identify simple variation (SNPs, insertions, deletions) leaving more complex variation (translocations, inversions) unexamined.

Genomic variation such as translocations and inversions have been shown to cause many human diseases. Translocations have been shown to be the cause of several different types of cancer, such as Burkitt's lymphoma [1] and acute promyelocytic leukemia [**?**]. They have also been shown to be associated with schizophrenia [2]. Studying and identifying different types of genomic polymorphisms could have impact on two very important fields in biology: genome wide association studies as well as phylogenetic tree reconstruction.

A. Genome Wide Association Studies

Commonly, in genome wide association studies (GWAS), next-generation sequence (NGS) reads are mapped to a reference genome. Differences, commonly SNPs and indels, are then identified from the read mapping results. This method has helped identify and associate many mutations with different diseases.

Read mapping, however, is a difficult task. More than 10% of reads were unmapped when mapping 12.2 million reads to the human genome using the popular Burrows-Wheeler Aligner [3]. Some of the reads will be left unmapped due to errors generated during sequencing. Other reads are left unmapped for unknown reasons. It may be that some unmapped reads vary significantly from the reference genome making read mapping difficult.

Mapped reads represent reads that are similar enough to the reference genome to be mapped with a given set of parameters. Unmapped reads may contain more interesting and novel biological information than mapped reads because these reads diverge enough from the reference genome to remain unmapped. Harnessing unmapped reads enables more thorough analysis of how individuals within a species differ and how genomic rearrangements may affect phenotypes.

B. Phylogenetic Tree Reconstruction

Phylogenetic tree reconstruction is often completed through comparing homologous gene sequences in a group of species of interest. Identification of homologous genes is a difficult task and is often a conservative process, allowing for only gene sequences that are very similar to be clustered together [4]. This approach is limited because it only allows for comparing gene sequences instead of comparing whole genomes [5]. Comparing the entire genome of one species to another is valuable to see if genomic rearrangements or other structural variations occurred to the genome. Accounting for

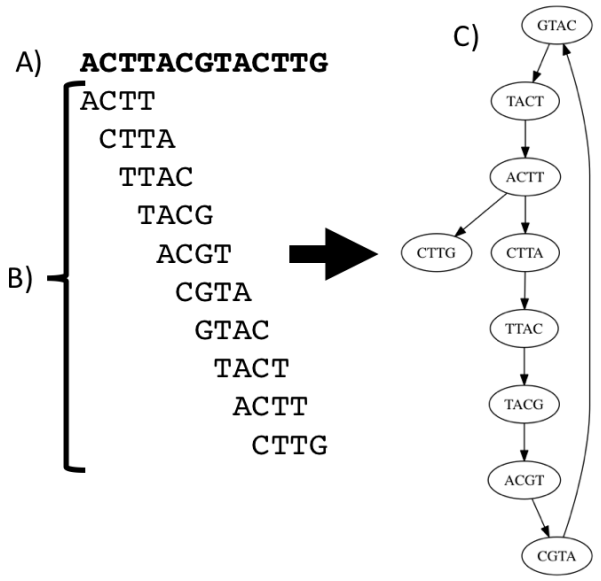

Figure 1. The construction of a standard de Bruijn Graph. **A**) The original sequence. **B**) sequence broken into kmers (k=4) showing kmer overlap. **C**) A de Bruijn graph with edges formed from overlapping kmers.

these genomic variations may serve as a future phylogenetic signal in future phylogenetic tree reconstruction.

II. Methods

Our method for utilizing unmapped reads and to compare whole genomes is to construct a relaxed de Bruijn graph that allows for more complex genomic variation to be observable.

A. Standard de Bruijn Graph

A standard de Bruijn graph is a graph structure that represents the genome of an organism. de Bruijn graphs are usually representative of a single species and are commonly used for genome assembly [3], [6]. Beyond genome assembly, they have also been found to increase the percent mapped reads when mapping reads to a de Bruijn graph versus contigs [7].

In a de Bruijn graph, each node represents a unique kmer. Edges in the graph represent kmer overlaps. The graph is usually constructed from NGS reads where reads are broken into kmers and used to populate the graph (see Figure 1).

B. Relaxed de Bruijn Graph

Our relaxed de Bruijn graph differs from a standard de Bruijn Graph is two major ways:

1) The graph contains sequence information for multiple species
2) Kmers can occur multiple times in the graph

By relaxing these constraints on the de Bruijn graph, we are able to identify interesting genomic variation in a tractable amount of time and space. Conceptually, this method can be thought of as merging two separate de Bruijn graphs by exploiting uniquely occurring kmers in one sequence as anchor points to merge the graphs.

Algorithm 1 Initial relaxed de Bruijn graph construction.

procedure CONSTRUCT(seq, k)
  Input: DNA sequence *seq*, kmer length k
  Output: kmer counts *occs*,
    index counter *curidx*,
    relaxed de Bruijn graph g
    kmer-index reverse lookup table *rlookup*
  occs ← occurrences of each kmer
  curidx ← 0
  g ← an empty graph
  for each kmer *kmer* in seq **do**
    l ← prefix of *kmer*
    lidx ← *curidx*
    curidx ← *curidx* + 1
    occs[l] ← *occrs*[l] + 1
    rlookup[l] ← *lidx*
    r ← suffix of *kmer*
    ridx ← *curidx*
    curidx ← *curidx* + 1
    occs[r] ← *occrs*[r] + 1
    rlookup[r] ← *ridx*
    $g.addedge(lidx, ridx)$
  end for
  return $occs, curidx, g$
end procedure

1) Graph Construction: Our graph construction algorithm is outlined in Algorithm 1, also see Figure 2 for a visual representation of the graph construction process. After graph construction, we simplify the graph by collapsing neighboring nodes in a graph where that path through the nodes is unambiguous to form unitigs.

2) Implementation: The NetworkX python package [8] was used for storing and manipulating de Bruijn graphs, Gephi [9] and Graphviz [10] were used for graph visualization.

III. Results and Discussion

A. Mapping Synthetic Reads

We created a synthetic genome (6930 base pairs) from and inserted the following polymorphisms:

1) Mutation (position 200)
2) Insertion (position 300, 9 base pairs long)
3) Inversion (position 400, 75 base pairs long)
4) Translocation (position 1000, 50 base pairs long originating from position 600)

150 base pair reads were simulated using ART Illumina [11] at 10x coverage using default settings with no errors and only in the forward direction. We generated a relaxed de Bruijn graph from the original 6930 bps reference sequence

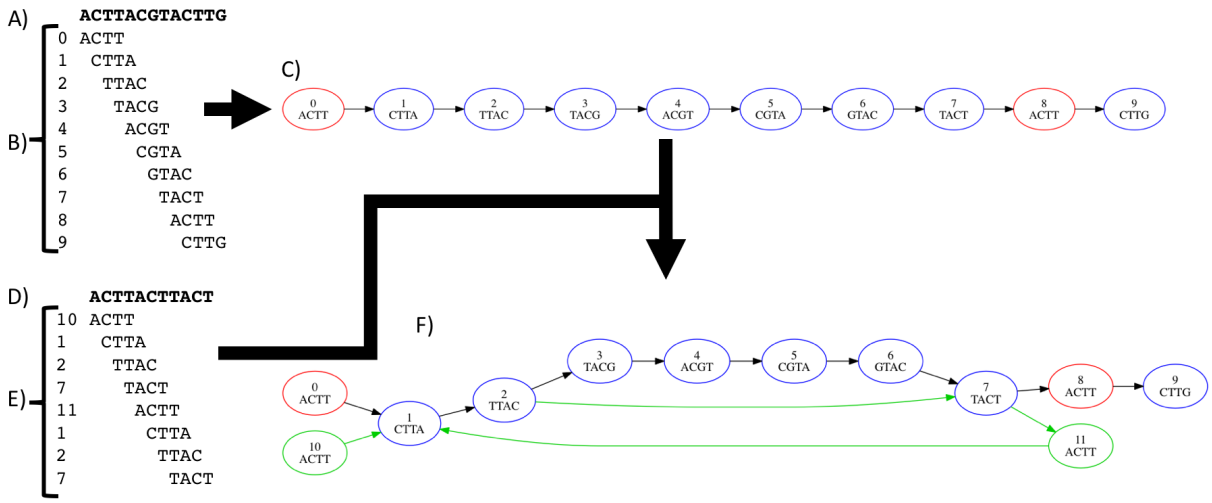

Figure 2. Construction method for our relaxed de Bruijn Graph for two reference genome sequences. **A** and **D** are two different sequences. **B** and **E** represent the sequence broken into kmers and the graph node IDs assigned to each kmer. **C** is the initial relaxed de Bruijn graph containing only **A**. Blue nodes are unique kmers and red nodes are non-unique kmers occurring in sequence **A**. **F** is the resulting relaxed de Bruijn graph once kmers from sequence **D** are added. Green nodes and edges are new nodes or edges that were added to the graph. See Algorithm 1 for construction of **C** and Algorithm 2 for **F**.

Algorithm 2 Appending new sequences after initial graph construction.

procedure APPEND($g, occs, curidx, seq, k, rlookup$)
  Input: initialized relaxed de Bruijn graph g
    kmer occurrence counter *occs*
    index counter *curidx*
    DNA sequence *seq*
    kmer length k
    kmer-index reverse lookup table *rlookup*
  for each kmer *kmer* in seq **do**
    $l \leftarrow$ prefix of *kmer*
    if $occs[l] == 1$ **then**
      $lidx \leftarrow rlookup[l]$
    else
      $lidx \leftarrow curidx$
      $curidx \leftarrow curidx + 1$
    end if
    $r \leftarrow$ suffix of *kmer*
    if $occs[r] == 1$ **then**
      $lidx \leftarrow rlookup[r]$
    else
      $ridx \leftarrow curidx$
      $curidx \leftarrow curidx + 1$
    end if
    $g.addedge(lidx, ridx)$
  end for
end procedure

and these reads using $k = 31$. The generated graph after unitig simplification can be seen in Figure 3.

In the simplified graph, the mutation, insertion and inversion form simple bubble structures (graph structure where a node has multiple outgoing edges to other nodes that later merge as incoming edges into another node) in the graph while the translocation forms a much more complex structure. In these very ideal conditions (all kmers in the reference sequence are unique, reads with no errors), the generated graph shows structures that could be used to generated a phylogenetic signal or for phenotype association with additional generated graphs from other individuals.

B. Comparing Real Whole Genomes

We compared two real *Escherichia coli* (strain K12) genomes that are very similar. We used *E. coli* K12/MG1655 (U00096.3) and K12/W3110 (NC_007779.1).

Using $k = 1001$, we generated a relaxed de Bruijn graph shown in Figure 4. Even with the extremely large k, there are still repeated kmers in the graph. Cycles caused by the repeated kmers can be seen in the graph. The graph constructed from real data where repeats occur is much more convoluted compared to the synthetic graph.

IV. Conclusion and Future Work

In this work, we have presented a method for constructing a unified, relaxed de Bruijn graph that contains more than one sequence source. The relaxed de Bruijn graph enables identification of graph structures that may be used as a signal for phylogenetic tree reconstruction or for use in association studies for phenotypes.

In the future, we plan to augment our algorithm in several ways: to be sensitive to sequencing errors as well

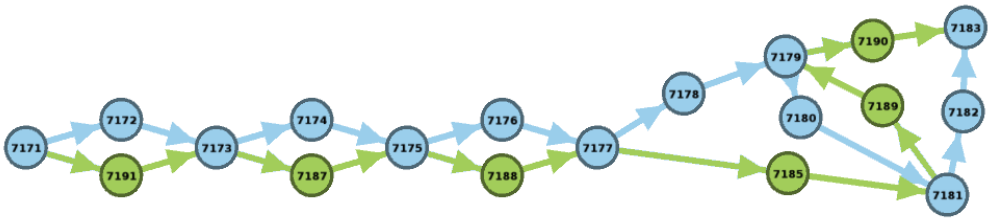

Figure 3. Graph structure formed from a synthetic genome and synthetically generated reads after node simplification to unitigs. Graph was constructed using $k = 31$. Blue nodes are sequence from the reference genome and green are sequences from synthetically generated NGS reads. Node 7191 contains a point mutation, node 7187 contains an insertion, node 7188 contains an inversion and the structure that forms from nodes 7185, 7189, and 7190 represent a translocation.

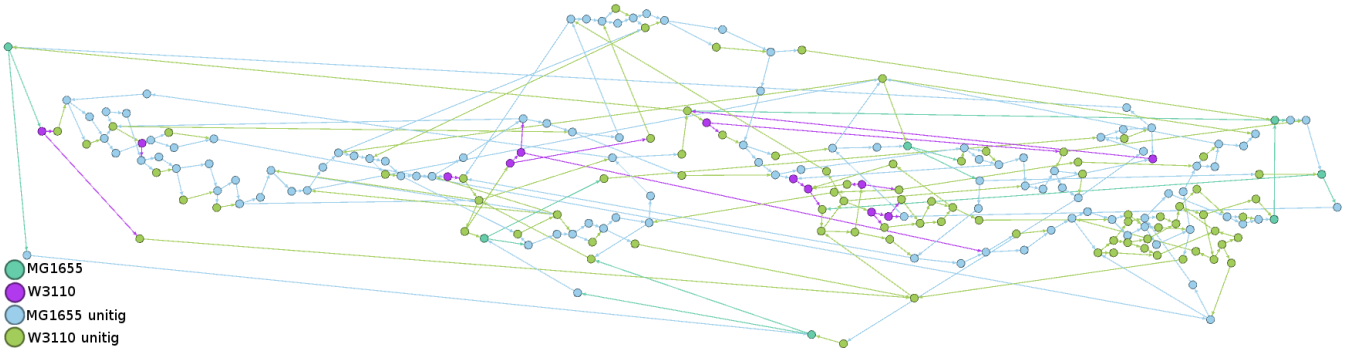

Figure 4. Relaxed de Bruijn graph for two *E. coli* genomes (MG1655 and W3110) using $k = 1001$. Many graph structures appear beyond simple bubbles. Additional methods are needed for characterization of complex graph structures that form.

as sequenced reads from the reverse direction, be resilient to repeat kmers, remove uninformative bubbles that form from the graph construction process, and identify complex graph structures that form.

Acknowledgment

This work was funded through the Utah NASA Space Grant Consortium and EPSCoR and through the BYU Graduate Research Fellowship.

The authors would like to thank J. Andrew Jacobsen, Paul M. Bodily, Justin Miller, Brandon Pickett, Sage Wright and Michael Cormier for their help and insight during this project.

References

[1] R. Hoffman, E. J. Benz Jr, L. E. Silberstein, H. Heslop, J. Anastasi, and J. Weitz, *Hematology: basic principles and practice*. Elsevier Health Sciences, 2013.

[2] J. E. Eykelenboom, G. J. Briggs, N. J. Bradshaw, D. C. Soares, F. Ogawa, S. Christie, E. L. Malavasi, P. Makedonopoulou, S. Mackie, M. P. Malloy *et al.*, "A t (1; 11) translocation linked to schizophrenia and affective disorders gives rise to aberrant chimeric disc1 transcripts that encode structurally altered, deleterious mitochondrial proteins," *Human molecular genetics*, vol. 21, no. 15, pp. 3374–3386, 2012.

[3] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[4] M. S. Fujimoto, A. Suvorov, N. O. Jensen, M. J. Clement, Q. Snell, and S. M. Bybee, "The ogcleaner: filtering false-positive homology clusters," *Bioinformatics*, vol. 33, no. 1, pp. 125–127, 2016.

[5] L. Li, C. J. Stoeckert, and D. S. Roos, "Orthomcl: identification of ortholog groups for eukaryotic genomes," *Genome research*, vol. 13, no. 9, pp. 2178–2189, 2003.

[6] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu *et al.*, "Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler," *Gigascience*, vol. 1, no. 1, p. 18, 2012.

[7] A. Limasset, B. Cazaux, E. Rivals, and P. Peterlongo, "Read mapping on de bruijn graphs," *BMC bioinformatics*, vol. 17, no. 1, p. 237, 2016.

[8] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Laboratory (LANL), Tech. Rep., 2008.

[9] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154

[10] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *SOFTWARE - PRACTICE AND EXPERIENCE*, vol. 30, no. 11, pp. 1203–1233, 2000.

[11] W. Huang, L. Li, J. R. Myers, and G. T. Marth, "Art: a next-generation sequencing read simulator," *Bioinformatics*, vol. 28, no. 4, pp. 593–594, 2011.