splitMEM: graphical pan-genome analysis with suffix skips

Shoshana Marcus
May 7, 2014
Outline

1  Overview
2  Data Structures
3  splitMEM Algorithm
4  Pan-genome Analysis
Objective

Input

A
B
C
D

Output

Compressed de Bruijn graph

- Graphical representation depicts how population variants relate to each other, especially where they diverge at branch points
- How well conserved is a sequence?
- What are network properties?

• Several complete genomes
• Available today for many microbial species, near future for higher eukaryotes
• Pan-genome: analyze multiple genomes of species together
de Bruijn graph

- Node for each distinct kmer
- Directed edge connects consecutive kmers
- Nodes overlap by k-1 bp
- Self-loops, multi-edges

Reconstruct original sequence:
Eulerian path through graph, visit each edge once
Compressed de Bruijn graph

- Merge non-branching chains of nodes
- Min. number of nodes that preserve path labels

*Usually built from uncompressed graph*
*We build directly in $O(n \log n)$ time and space*
Compressed de Bruijn graph

9 strains of *Bacillus anthracis* \( k=25 \)
Compressed de Bruijn graph

9 strains of *Bacillus anthracis* \( k=1000 \)
Outline

1  Overview

2  Data Structures

3  splitMEM Algorithm

4  Pan-genome Analysis
Suffix Tree

• Rooted, directed tree with leaf for each suffix.
• Each internal node, except the root, has at least two children.
• Each edge is labeled with nonempty substring.
• No two siblings begin with the same character.
• Path from root to leaf $i$ spells suffix $S[i \ldots n]$.
• Append special character $\$\$ to guarantee each suffix ends at leaf.
Constructing Suffix Tree

Naïve Algorithm

$S = \text{banana}\$

$suf_1$

banana

banana$
Constructing Suffix Tree

Naïve Algorithm

$S = \text{banana}\$

$suf_1$

$suf_2$

banana$\$

$\$eanae$
Constructing Suffix Tree

Naïve Algorithm

\[ S = \text{banana}\$ \]
Constructing Suffix Tree

Naïve Algorithm

S = banana$

suf_2
suf_3
suf_1
suf_4
Constructing Suffix Tree

Naïve Algorithm

$S =$ banana$

suf$_2$

suf$_4$

suf$_1$

suf$_3$

suf$_4$

ana$
Constructing Suffix Tree

Naïve Algorithm

\[ S = \text{banana}\$ \]

- banana$  \rightarrow  \text{suf}_1
- anana$  \rightarrow  \text{suf}_2
- nana$  \rightarrow  \text{suf}_3
- ana$  \rightarrow  \text{suf}_4
- na$  \rightarrow  \text{suf}_5
- a$  \rightarrow  \text{suf}_6
- $  \rightarrow  \text{suf}_7

O(n^2) \text{ time}
Constructing Suffix Tree

O(n) time

On-line Constructin of Suffix Trees, E. Ukkonen
Algorithmica (1995)
Suffix Tree Query

\[ S = \text{banana}\$ \]

Search for \textbf{ban}
Suffix Tree Query

S = banana$

Search for ban
Suffix Tree Query

S = banana$

Search for ban
Suffix Tree Query

S = banana$

Search for ban

Found 1 occurrence
Suffix Tree Query

S = banana$

Search for band

Not found
Suffix Tree Query

\[ S = \text{banana}\$ \]

Search for an

Found 2 occurrences
Suffix Tree

- Many applications in computational biology
- Linear time construction algorithms

Linear time solutions to
  - Genome alignment
  - Finding longest common substring
  - All-pairs suffix-prefix matching
  - **Locating all maximal repetitions**
  - And many more…
MEMs

Maximal Exact Match (MEM)

Exact match within sequence that cannot be extended left or right without introducing mismatch.

We are interested in MEMs length $\geq k$
MEMs

Maximal Exact Match (MEM)

Exact match within sequence that cannot be extended left or right without introducing mismatch.

MEMs are internal nodes in the suffix tree that have left-diverse descendants.

(have descendant leaves that represent suffixes with different characters preceding them)

✧ Linear-time suffix tree traversal to locate MEMs.
MEMs in Suffix Tree

Possible MEMs: a, ana, na

\[ S = \text{banana}\$

\begin{align*}
\text{MEM?} & \quad \text{MEM?} & \quad \text{MEM?} \\
\text{banana}\$ & \quad \text{ana}\$ & \quad \text{na}\$
\end{align*}

MEMs are internal nodes in suffix tree with left-diverse descendants
MEMs in Suffix Tree

MEMs: a, ana

$S = \text{banana}$$\
\text{banana}$$ \quad \text{suf}_2
\text{nana}$$ \quad \text{suf}_4$

MEMs are internal nodes in suffix tree with left-diverse descendants
MEMs in Suffix Tree

MEMs: a, ana

$S = \text{banana}\$

\text{ana}a$\$ \text{suf}_3$

\text{ana}$ \text{suf}_5$

MEMs are internal nodes in suffix tree with left-diverse descendants
Outline

1 Overview
2 Data Structures
3 splitMEM Algorithm
4 Pan-genome Analysis
Compressed de Bruijn graph

Input:
AGAAGTCC
\$ATAAGTTA

Types of nodes:

i. repeatNodes
ii. uniqueNodes
splitMEM

Nodes in compressed de Bruijn graph classified as
i. repeatNodes
ii. uniqueNodes

Algorithm:
1. Construct set of repeatNodes
2. Sort start positions of repeatNodes
3. Create edges and uniqueNodes to link non-contiguous repeatNodes
repeatNodes

1. Construct set of repeatNodes
   1. Build suffix tree of genome
   2. Mark internal nodes that are MEMs, length $\geq k$
   3. Preprocess suffix tree for LMA queries

4. Compute repeatNodes in compressed de Bruijn graph by decomposing MEMs and extracting overlapping components, length $\geq k$
1 MEM occurs twice
Overlapping MEMs

TGCCATCGCCAAACCAT

TGCCATCGCCAAACCAT
Tandem Repeat

AGGCTTTGGGCTTTGGGCTTGGGCTA
AGGCTTTGGGCTTTGGGCTTGGGCTA
AGGCTTTGGGCTTTGGGCTTGGGCTA
AGGCTTTGGGCTTTGGGCTTGGGCTA
AGGCTTTGGGCTTTGGGCTTGGGCTA

GGCT
CTTGGG
repeatNodes

1. Construct set of repeatNodes
   1. Build suffix tree of genome
   2. Mark internal nodes that are MEMs, length $\geq k$
   3. Preprocess suffix tree for LMA queries
   4. Compute repeatNodes in compressed de Bruijn graph by decomposing MEMs and extracting overlapping components, length $\geq k$
Split MEM to repeatNodes
Split MEM to repeatNodes

Find MEM in suffix tree.
Split MEM to repeatNodes

Traverse suffix link.
Look for MEM as ancestor.
Split MEM to repeatNodes

Traverse suffix link.
Look for MEM as ancestor.
Split MEM to repeatNodes

Traverse suffix link.
Look for MEM as ancestor.
Split MEM to repeatNodes

Found MEM as ancestor. Decompose.
Remove embedded MEM (suffix links). Find next embedded MEM.
Suffix Skips

- Reduce $O(n^2)$ time to $O(n \log n)$ time

Suffix link: quickly navigate to distant part of tree
- Pointer from internal node labeled $xS$ to node $S$
- Trim 1 character in $O(1)$ time
- Trim $c$ characters in $O(c)$ time

Suffix skip:
- Trim $c$ characters in $O(\log c)$ time
Suffix Skips

Genome: babab

Suffix skips 0
(dist = 1; suffix links)

Suffix skips 1
(dist=2)

Suffix skips 2
(dist=4)

Additional Preprocessing:
pointer jumping to rapidly add additional links
splitMEM

• splitMEM software
  o C++
  o open source http://splitmem.sourceforge.net

• Input modes:
  o single genome: fasta file
  o pan-genome: multi-fasta file

• Multi k-mer
  construct several compressed de Bruijn graphs without rebuilding suffix tree
Outline

1. Overview
2. Data Structures
3. splitMEM Algorithm
4. Pan-genome Analysis
Pan-genome analysis

*B. Anthracis* and *E. coli*

Examine graph properties:
- Number nodes, edges, avg. degree
- Node length distribution
- Genome sharing among nodes
- Distribution of node distances to core genome

Other properties that can be studied:
- Girth, Diameter, Modularity, Network Motifs, etc.
- Functional enrichment of highly conserved or genome specific genes.
Pan-genome analysis

<table>
<thead>
<tr>
<th>Strain</th>
<th>Size (KB)</th>
<th>Accession</th>
</tr>
</thead>
<tbody>
<tr>
<td>B. anthracis A0248 uid33543</td>
<td>5178</td>
<td>CP001598</td>
</tr>
<tr>
<td>B. anthracis A16R uid40353</td>
<td>5179</td>
<td>CP001974</td>
</tr>
<tr>
<td>B. anthracis A16 uid40303</td>
<td>5179</td>
<td>CP001970</td>
</tr>
<tr>
<td>B. anthracis Ames 0581 uid10784</td>
<td>5178</td>
<td>AE017334</td>
</tr>
<tr>
<td>B. anthracis Ames uid309</td>
<td>5178</td>
<td>AE016879</td>
</tr>
<tr>
<td>B. anthracis CDC 684 uid31329</td>
<td>5181</td>
<td>CP001215</td>
</tr>
<tr>
<td>B. anthracis Cl uid36309</td>
<td>5147</td>
<td>CP001746</td>
</tr>
<tr>
<td>B. anthracis H9401 uid49361</td>
<td>5170</td>
<td>CP002091</td>
</tr>
<tr>
<td>B. anthracis str Sterne uid10878</td>
<td>5180</td>
<td>AE017225</td>
</tr>
<tr>
<td>E. coli 0127 H6 E2348 69 uid32571</td>
<td>4919</td>
<td>FM180568</td>
</tr>
<tr>
<td>E. coli 042 uid40647</td>
<td>5193</td>
<td>FN554766</td>
</tr>
<tr>
<td>E. coli 536 uid16235</td>
<td>4893</td>
<td>CP000247</td>
</tr>
<tr>
<td>E. coli 55989 uid33413</td>
<td>5107</td>
<td>CU928145</td>
</tr>
<tr>
<td>E. coli ABU 83972 uid38725</td>
<td>5083</td>
<td>CP001671</td>
</tr>
<tr>
<td>E. coli APEC O1 uid16718</td>
<td>5034</td>
<td>CP000468</td>
</tr>
<tr>
<td>E. coli APEC O78 uid184588</td>
<td>4753</td>
<td>CP004009</td>
</tr>
<tr>
<td>E. coli BL21 DE3 uid20713</td>
<td>4516</td>
<td>CP001509</td>
</tr>
<tr>
<td>E. coli BL21 DE3 uid28965</td>
<td>4516</td>
<td>AM946981</td>
</tr>
</tbody>
</table>
Pan-genome analysis

Graphs of main chromosomes
• 9 strains of *Bacillus anthracis*
• Selection of 9 strains of *Escherichia coli*

<table>
<thead>
<tr>
<th>Species</th>
<th>K</th>
<th>Nodes</th>
<th>Edges</th>
<th>Avg. Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>B. anthracis</td>
<td>25</td>
<td>103926</td>
<td>138468</td>
<td>1.33</td>
</tr>
<tr>
<td>B. anthracis</td>
<td>100</td>
<td>41343</td>
<td>54954</td>
<td>1.32</td>
</tr>
<tr>
<td>B. anthracis</td>
<td>1000</td>
<td>6627</td>
<td>8659</td>
<td>1.30</td>
</tr>
<tr>
<td>E. coli</td>
<td>25</td>
<td>494783</td>
<td>662081</td>
<td>1.33</td>
</tr>
<tr>
<td>E. coli</td>
<td>100</td>
<td>230996</td>
<td>308256</td>
<td>1.33</td>
</tr>
<tr>
<td>E. coli</td>
<td>1000</td>
<td>11900</td>
<td>15695</td>
<td>1.31</td>
</tr>
</tbody>
</table>
Pan-genome analysis

*B. Anthracis* and *E. coli*

Examine graph properties

- **Node length distribution**
- Genome sharing among nodes
- Distribution of node distances to core genome
Pan-genome analysis

Histogram of Node Lengths
Pan-genome analysis

Histogram of Node Lengths

Node Length Histogram for B. anthracis
mean = 382 ± 4501 max = 45

Spike at 2k: SNPs
Pan-genome analysis

*B. Anthracis* and *E. coli*
Examine graph properties
• Node length distribution
• Genome sharing among nodes
• Distribution of node distances to core genome
Pan-genome analysis

Fraction of nodes with each level of genome sharing

B. anthracis

E. coli
Pan-genome analysis

*B. Anthracis* and *E. coli*

Examine graph properties

- Node length distribution
- Genome sharing among nodes
- Distribution of node distances to core genome
Pan-genome analysis

Graph encodes sequence context of segments.

Core genome: subsequences that occur in at least 70% of underlying genomes.

Nodes can be further in terms of hops while closer by base pairs.

Branch and Bound Search
Pan-genome analysis

Node distances to core genome

Searched 1000-hop radius
Summary

• Identify pan-genome relationships graphically.

• Topological relationship between suffix tree and compressed de Bruijn graph.

• Direct construction of compressed de Bruijn graph for single or pan-genome.

• Introduce suffix skips.

• Explore pan-genome graphs of *B. anthracis*, *E. coli*.

*SplitMEM*: Graphical pan-genome analysis with suffix skips.


http://biorxiv.org/content/early/2014/04/06/003954
Future work

Improve splitMEM software:
• Reduce space using compressed full-text index instead of suffix tree
• Approximate indexing of strains to form a pan-genome graph
• Alignment of reads to pan-genome

Biological applications:
• Functional enrichment of core-genome and genome specific segments
• Expand study to larger collection of microbes and larger genomes
Acknowledgments

Michael Schatz
Hayan Lee
Giuseppe Narzisi
James Gurtowski
Schatz Lab

IT department
Todd Heywood

iPlant Collaborative™
Thank You!
Pan-genome analysis

Branch and bound search (like Dijkstra’s shortest path algorithm) to compute bp distance from each non-core node to core genome:

Traverse all distinct paths from source until
  o a core node is reached
OR
  o current node was visited by a shorter path

Bounded search

once a core node is found, its distance bounds maximum search distance along other paths