Methods for Indexing and Searching Large-Scale Genomic Data

Prashant Pandey
ppandey@berkeley.edu
Berkeley Lab/UC Berkeley
The cost of sequencing is going down

The cost of sequencing human genome is going down over years
SRA contains a lot of *diversity information*

Q: What if I find e.g., a new disease-related gene, and want to see if it appeared in other experiments?
New challenges due to data growth

SRA contains a lot of *diversity information*

Only a small portion of SRA is searchable!

This renders what is otherwise an immensely valuable public resource *largely inert*
This new regime, in which costs scale with the amount of computational processing time, places a premium on driving down the average cost by developing efficient algorithms for data processing.”
The computational challenge

“This new regime, in which costs scale with the amount of computational processing time, places a premium on driving down the average cost by developing efficient algorithms for data processing.”

Also, it’s not just “new” data that is the problem

In addition to new data, re-analysis of existing experiments often desired: In light of new annotations, discoveries, and methodological advancements.
Three approaches to handle massive data

**Goal:** make data smaller to fit in RAM

**Techniques:**
- LSH e.g., MinHash
- Filters, e.g., Bloom filter
- Succinct data structures
Three approaches to handle massive data

Shrink it

**Goal:** make data smaller to fit in RAM

**Techniques:**
- LSH e.g., MinHash
- Filters, e.g., Bloom filter
- Succinct data structures

Organize it

**Goal:** organize data in a disk-friendly way

**Techniques:**
- B-tree
- Bε-tree
- LSM-tree
Three approaches to handle massive data

**Shrink it**

**Goal:** make data smaller to fit in RAM

**Techniques:**
- LSH e.g., MinHash
- Filters, e.g., Bloom filter
- Succinct data structures

**Organize it**

**Goal:** organize data in a disk-friendly way

**Techniques:**
- B-tree
- B⁺-tree
- LSM-tree

**Distribute it**

**Goal:** partition and distribute data on multiple nodes

**Techniques:**
- Distributed hash table
- Distributed key-value store
Our solutions to handle massive data

- **Shrink it**
  - (Counting) Quotient Filter
    - SIGMOD ‘17, arXiv ‘17
  - Order Min Hash
    - ISMB ‘19

- **Organize it**
  - Buffered Count-Min Sketch
    - ESA ‘18
  - Affine & PDAM model
    - SPAA ‘19

- **Distribute it**
Our solutions to handle massive data

**Shrink it**
- **(Counting) Quotient Filter**
  - SIGMOD ‘17, arXiv ‘17
- **Order Min Hash**
  - ISMB ‘19
- **Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis**
  - ISMB ‘17, WABI ‘17, BIOINFORMATICS ‘17, RECOMB ‘18, Cell Systems ‘18, RECOMB ‘19, JCB ‘20

**Organize it**
- **Buffered Count-Min Sketch**
  - ESA ‘18
- **Affine & PDAM model**
  - SPAA ‘19
- **LSM-Mantis, VaraintStore**
  - bioRxiv ‘20, bioRxiv ‘21

**Distribute it**
- **Distributed GPU-based k-mer counting**
  - IPDPS ‘21
Our solutions to handle massive data

Shrink it

- (Counting)
- Quotient Filter
- SIGMOD ‘17, arXiv ‘17
- Order Min Hash
- ISMB ‘19

Organize it

- Buffered Count-Min Sketch
- ESA ‘18
- Affine & PDAM model
- SPAA ‘19
- LSM-Mantis, VaraintStore
- bioRxiv ‘20, bioRxiv ‘21

Distribute it

- Distributed GPU-based k-mer counting
- IPDPS ‘21

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
- ISMB ‘17, WABI ‘17, BIOINFORMATICS ‘17, RECOMB ‘18, Cell Systems ‘18, RECOMB ‘19, JCB ‘20
- Buffered Count-Min Sketch
- ESA ‘18
- Affine & PDAM model
- SPAA ‘19
- LSM-Mantis, VaraintStore
- bioRxiv ‘20, bioRxiv ‘21

BɛtrFS file system
- FAST ‘15, TOS 15, FAST ‘16, TOS 16

Distributed GPU-based k-mer counting
- IPDPS ‘21
Our solutions to handle massive data

**Shrink it**

- Quotient Filter
  - SIGMOD ‘17, arXiv ‘17
- Order Min Hash
  - ISMB ‘19

**Organize it**

- Buffered Count-Min Sketch
  - ESA ‘18
- Affine & PDAM model
  - SPAA ‘19

- LSM-Mantis, VaraintStore
  - bioRxiv ‘20, bioRxiv ‘21

- BétrFS file system
  - FAST ‘15, TOS 15, FAST ‘16, TOS 16

**Distribute it**

- Distributed GPU-based k-mer counting
  - IPDPS ‘21

**Related Works**

- Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
  - ISMB ‘17, WABI ‘17, BIOINFORMATICS ‘17, RECOMB ‘18, Cell Systems ‘18, RECOMB ‘19, JCB ‘20

- LERTs (Event reporting)
  - arXiv ‘19, SIGMOD ‘20
In this talk: Order Min Hash (OMH)

Shrink it

(Counting) Quotient Filter
SIGMOD '17, arXiv '17

Order Min Hash
ISMB '19

Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis
ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20

Organize it

Locality-sensitive hashing for the edit distance
Guillaume Marçais*, Dan DeBlasio, Prashant Pandey and Carl Kingsford*
Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Distribute it

Affine & PDAM model
SPAA '19

LSM-Mantis, VaraintStore
bioRxiv '20, bioRxiv '21

BεtrFS file system
FAST '15, TOS 15, FAST '16, TOS 16

LERTs (Event reporting)
arXiv '19, SIGMOD '20

Buffered Count-Min Sketch
ESA '18

IPDPS '21

Bioinformatics, 35, 2019, i127–i135
doi: 10.1093/bioinformatics/btz284
ISMB/ECCB 2019
Sequence similarity problem

Sequence similarity is a measure of the similarity of two sequences. Eg., Edit distance between two sequences is a measure of their similarity.

Low edit distance $\Leftrightarrow$ High similarity
High edit distance $\Leftrightarrow$ Low similarity

Measuring sequence similarity is the core problem in many algorithms in computational biology

- Genome assembly (overlap-layout-consensus) [Jaffe et al. 2003, Myers et al. 2000]
Sequence similarity is a measure of the similarity of two sequences. Eg., Edit distance between two sequences is a measure of their similarity.

Low edit distance ⇔ High similarity

Computing quadratic-time edit distance between sequences at scale is computationally not feasible in practice!

- Genome assembly (overlap-layout-consensus) [Jaffe et al. 2003, Myers et al. 2000]
Overlap computation

- Compute overlaps between reads
Overlap computation

- Compute overlaps between reads
- Instance of “Nearest Neighbor Problem” for edit distance
- Compute overlaps between reads
- Instance of “Nearest Neighbor Problem” for edit distance
- Use multiple hash tables

Overlap computation

Hash tables
Overlap computation

- Compute overlaps between reads
- Instance of “Nearest Neighbor Problem” for edit distance
- Use multiple hash tables
- Need meaningful hash collisions
Locality Sensitive Hashing (LSH)

Pick $h$ at random from $\mathcal{H}$:

$$\Pr[h(\bigcirc) = h(\bigcirc)] > \Pr[h(\bigcirc) = h(\triangle)]$$

**Locality sensitive hash family**

Family of hash functions where similar elements are more likely to have the same value than distant elements.
Locality Sensitive Hashing (LSH)

**Locality sensitive hash family**
Family of hash functions where similar elements are more likely to have the same value than distant elements.

Pick \( h \) at random from \( \mathcal{H} \):

\[
\Pr[h(\bigcirc) = h(\bigcirc)] > \Pr[h(\bigcirc) = h(\triangle)]
\]

Sketch (\( \bigcirc \)) = \{\( h_1(\bigcirc), \ldots, h_m(\bigcirc) \}\)
The family $\mathcal{H}$ is sensitive for distance $D$ such that for all $x, y \in U$

$$\Pr[h(x) = h(y)] = 1 - D(x, y)$$

**Locality sensitive hash family**

Family of hash functions where similar elements are more likely to have the same value than distant elements.
The family $\mathcal{H}$ is sensitive for distance $D$ if there exists $d_1 < d_2, p_1 > p_2$ such that for all $x, y \in U$

$$D(x, y) < d_1 \Rightarrow \Pr[h(x) = h(y)] \geq p_1$$

$$D(x, y) \geq d_2 \Rightarrow \Pr[h(x) = h(y)] \leq p_2$$

- Low distance $\Leftrightarrow$ High collisions
- High distance $\Leftrightarrow$ Low collisions

**Locality sensitive hash family**

Family of hash functions where similar elements are more likely to have the same value than distant elements.
Jaccard distance $\rightarrow$ proxy for edit distance

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$
Jaccard distance → proxy for edit distance

Jaccard distance between sequences $x$, $y$: Jaccard distance of their $k$-mer sets

\[ J(x, y) = J(\mathcal{K}(x), \mathcal{K}(y)) \]

\[ J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \]
Jaccard distance → proxy for edit distance

Jaccard distance between sequences $x, y$:
Jaccard distance of their $k$-mer sets

$$J(x, y) = J(\mathcal{K}(x), \mathcal{K}(y))$$

- Low $D(x, y) \Rightarrow$ Low $J(x, y)$
- High $D(x, y) \not\Rightarrow$ High $J(x, y)$

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$
Mash extends the minHash dimensionality-reduction technique to include a pairwise mutation distance and $P$ value significance test, enabling the efficient *clustering* and *search* of massive sequence collections.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{|S(A \cup B) \cap S(A) \cap S(B)|}{|S(A \cup B)|}$$
Jaccard ignore $k$-mer repetition

\[
x = \text{AAAAAA} \text{AAAAAAAAAA} \text{CCCC}
\]
\[
y = \text{AAAAA} \text{CCCCCCCCCCCCCCCC}
\]
Jaccard ignore $k$-mer repetition

\[ x = \text{AAAAAAAAAAAAAAAAAAAAAAAACCCC} \rightarrow \text{AAAAA, AAAAC, AAACC, AACCC, ACCCC, CCCCC} \]

\[ y = \text{AAAAAACCCCCCCCCCCCCCCCCCC} \rightarrow \text{AAAAA, AAAAC, AAACC, AACCC, ACCCC, CCCCC} \]
Jaccard distance $J(x, y) = 0$  Edit distance $D(x, y) \geq 1 - 2k/n$

Identical $k$-mer content but high edit distance
Weighted Jaccard handles repetitions

Generalized Jaccard distance for multi-sets

\[ J^W(A, B) = 1 - \frac{\sum_{x \in U} \min(x_A(x), x_B(x))}{\sum_{x \in U} \max(x_A(x), x_B(x))} \]
Weighted Jaccard handles repetitions

Jaccard distance $J^w(x, y) = 1 - (k+2)/n$  Edit distance $D(x, y) \geq 1 - 2k/n$

Weighted Jaccard is closer to edit distance than Jaccard
Jaccard and weighted Jaccard ignore relative order

\[ x = \text{CCCCACCAACACAAAAACCC} \]

\[ y = \text{AAAAACACAAACCCCCACCAAA} \]

\( x, y: \) de Bruijn sequences, contain all 16 possible 4-mers once
Jaccard and weighted Jaccard ignore relative order

\[ J(x, y) = J^w(x, y) = 0 \quad D(x, y) = 0.63 \]

\( x = \text{CCCCACCAACACAAACCC} \quad \rightarrow \quad \text{AAAA, AAAC, ACAA, AAAC, ACAC, ACCA, ACCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, CCCA, CCC} \)

\( y = \text{AAAAACACAACCCCCACCCAAA} \quad \rightarrow \quad \text{AAAA, AAAC, AACA, AAC, ACAC, ACCA, ACCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, CCCA, CCC} \)

\( x, y: \text{de Bruijn sequences, contain all 16 possible 4-mers once} \)
OMH: Order Min Hash

- minHash is an LSH for Jaccard
- OMH is a refinement of minHash
- OMH is sensitive to
  - repeated $k$-mers
  - relative order of $k$-mers
$x = \text{AGTTGAGCGGAAGGTG}$  $k = 2$

Order: permutation of $\Sigma^k$

```
AT
AG
GT
CG
GT
TT
GG
CA
TG
TC
AA
AC
TA
GA
GC
CT
```
minHash and OMH sketch

\[ x = \text{AGTTGAGCGGAAGGTG} \quad k = 2 \quad m = 6 \]

Order: permutation of \( \Sigma^k \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AG</td>
<td>GG</td>
<td>CG</td>
<td>AA</td>
<td>TG</td>
<td>TT</td>
</tr>
<tr>
<td>2</td>
<td>GT</td>
<td>GA</td>
<td>GA</td>
<td>AG</td>
<td>TT</td>
<td>GG</td>
</tr>
<tr>
<td>3</td>
<td>CG</td>
<td>CG</td>
<td>TG</td>
<td>TT</td>
<td>GG</td>
<td>AG</td>
</tr>
<tr>
<td>4</td>
<td>TT</td>
<td>AG</td>
<td>AG</td>
<td>GT</td>
<td>CG</td>
<td>GC</td>
</tr>
<tr>
<td>5</td>
<td>GG</td>
<td>GC</td>
<td>GC</td>
<td>TG</td>
<td>AA</td>
<td>GA</td>
</tr>
<tr>
<td>6</td>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GC</td>
<td>GT</td>
<td>AA</td>
</tr>
<tr>
<td>7</td>
<td>AA</td>
<td>AA</td>
<td>TT</td>
<td>GA</td>
<td>AG</td>
<td>CG</td>
</tr>
<tr>
<td>8</td>
<td>GA</td>
<td>TT</td>
<td>AA</td>
<td>CG</td>
<td>GC</td>
<td>TG</td>
</tr>
<tr>
<td>9</td>
<td>GC</td>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GA</td>
<td>GT</td>
</tr>
</tbody>
</table>
minHash and OMH sketch

\[ x = \text{AGTTGAGCGGAAGGTG} \]

\[ k = 2 \quad m = 6 \]

Order: permutation of \( \Sigma^k \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>GG</td>
<td>CG</td>
<td>AA</td>
<td>TG</td>
<td>TT</td>
<td></td>
</tr>
<tr>
<td>GT</td>
<td>GA</td>
<td>GA</td>
<td>AG</td>
<td>TT</td>
<td>GG</td>
<td></td>
</tr>
<tr>
<td>CG</td>
<td>CG</td>
<td>TG</td>
<td>TT</td>
<td>GG</td>
<td>AG</td>
<td></td>
</tr>
<tr>
<td>TT</td>
<td>AG</td>
<td>AG</td>
<td>GT</td>
<td>CG</td>
<td>GC</td>
<td></td>
</tr>
<tr>
<td>GG</td>
<td>GC</td>
<td>GC</td>
<td>TG</td>
<td>AA</td>
<td>GA</td>
<td></td>
</tr>
<tr>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GC</td>
<td>GT</td>
<td>AA</td>
<td></td>
</tr>
<tr>
<td>AA</td>
<td>AA</td>
<td>TT</td>
<td>GA</td>
<td>AG</td>
<td>CG</td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td>TT</td>
<td>AA</td>
<td>CG</td>
<td>GC</td>
<td>TG</td>
<td></td>
</tr>
<tr>
<td>GC</td>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GA</td>
<td>GT</td>
<td></td>
</tr>
</tbody>
</table>
$x = \text{AGTTGAGCGGAAGGTG} \quad k = 2 \quad m = 6$

Order: permutation of $\Sigma^k \times \{1, \ldots, n\}$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>GG</td>
<td>CG</td>
<td>AA</td>
<td>TG</td>
<td>TT</td>
</tr>
<tr>
<td>GT</td>
<td>GA</td>
<td>GA</td>
<td>AG</td>
<td>TT</td>
<td>GG</td>
</tr>
<tr>
<td>CG</td>
<td>CG</td>
<td>TG</td>
<td>TT</td>
<td>GG</td>
<td>AG</td>
</tr>
<tr>
<td>TT</td>
<td>AG</td>
<td>AG</td>
<td>GT</td>
<td>CG</td>
<td>GC</td>
</tr>
<tr>
<td>GG</td>
<td>GC</td>
<td>GC</td>
<td>TG</td>
<td>AA</td>
<td>GA</td>
</tr>
<tr>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GC</td>
<td>GT</td>
<td>AA</td>
</tr>
<tr>
<td>AA</td>
<td>AA</td>
<td>TT</td>
<td>GA</td>
<td>AG</td>
<td>CG</td>
</tr>
<tr>
<td>GA</td>
<td>TT</td>
<td>AA</td>
<td>CG</td>
<td>GC</td>
<td>TG</td>
</tr>
<tr>
<td>GC</td>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GA</td>
<td>GT</td>
</tr>
</tbody>
</table>

1

GA, 4
TG, 3
AG, 5
GT, 1
GT, 13
AA, 10
AG, 11
TT, 2
AG, 0
CG, 7
GG, 12
GC, 6
TG, 14
GG, 8
GA, 9
minHash and OMH sketch

\[ x = \text{AGTTGAGCGGAAGGTG} \quad k = 2 \quad m = 6 \]

Order: permutation of \( \Sigma^k \times \{1, \ldots, n\} \)

\[
\begin{array}{ccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
AG & GG & CG & AA & TG & TT \\
GT & GA & GA & AG & TT & GG \\
CG & CG & TG & TT & GG & AG \\
TT & AG & AG & GT & CG & GC \\
GG & GC & GC & TG & AA & GA \\
TG & GT & GG & GC & GT & AA \\
AA & AA & TT & GA & AG & CG \\
GA & TT & AA & CG & GC & TG \\
GC & TG & GT & GG & GA & GT \\
\end{array}
\]
minHash and OMH sketch

\[ x = \text{AGTTGAGCGGGAAGGTG} \quad k = 2 \quad m = 6 \quad l = 2 \]

Order: permutation of \( \Sigma^k \times \{1,\ldots, n\} \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AG</td>
<td>GG</td>
<td>CG</td>
<td>AA</td>
<td>TG</td>
<td>TT</td>
</tr>
<tr>
<td>2</td>
<td>GT</td>
<td>GA</td>
<td>GA</td>
<td>AG</td>
<td>TT</td>
<td>GG</td>
</tr>
<tr>
<td>3</td>
<td>CG</td>
<td>CG</td>
<td>TG</td>
<td>TT</td>
<td>GG</td>
<td>AG</td>
</tr>
<tr>
<td>4</td>
<td>TT</td>
<td>AG</td>
<td>AG</td>
<td>GT</td>
<td>GC</td>
<td>GA</td>
</tr>
<tr>
<td>5</td>
<td>GG</td>
<td>GC</td>
<td>GC</td>
<td>TG</td>
<td>AA</td>
<td>GA</td>
</tr>
<tr>
<td>6</td>
<td>TG</td>
<td>GT</td>
<td>GG</td>
<td>GC</td>
<td>GT</td>
<td>AA</td>
</tr>
</tbody>
</table>

\( x = \text{AGTTGAGCGGGAAGGTG} \)

\( k = 2 \)

\( m = 6 \)

\( l = 2 \)
The Jaccard similarity stays high even for sequences with high edit distance
OMH: conclusion

- an improvement over minHash
- easy to compute
- locality sensitive for edit distance
Country-scale sequencing efforts produce huge amount of variation data

- 1000 Genomes project [https://www.internationalgenome.org/]
- The Cancer Genome Atlas (TCGA) [https://portal.gdc.cancer.gov/]
- Genotype-Tissue Expression (GTEx) [https://gtexportal.org/home/]
Variation data analysis can improve downstream applications

- Population-level disease analysis
- Genome-wide association studies
- Personalized medicine
- Cancer remission-rate prediction
- Colocalization analysis
- PCR primer design
- Genome assembly
Variation data analysis can improve downstream applications

- Population-level disease analysis
- Genome-wide association studies
- Personalized medicine
- Cancer remission-rate prediction
- Colocalization analysis
- PCR primer design
- Genome assembly

Sequencing & assembly

 Individuals

Population Genomes

- Count the number of variants in a gene
- List all people, with > N variants in a gene
- Return all positions with variants in a gene
- List all people, with sequence S in a gene
- For person P, return the closest variant from position X
Multiple sample sequences and variants

<table>
<thead>
<tr>
<th>Sample 1</th>
<th>C A A T T T G C T G A T C T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 2</td>
<td>C A T G C T G A T C T</td>
</tr>
<tr>
<td>Sample 3</td>
<td>C G A T T T T G C T G A T C T</td>
</tr>
<tr>
<td>Sample 4</td>
<td>C G A T T T T A C G G C T G A T C T</td>
</tr>
</tbody>
</table>
Multiple sample sequences and variants

<table>
<thead>
<tr>
<th>Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample 1</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
</tr>
<tr>
<td>Sample 2</td>
<td>C</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample 3</td>
<td>C</td>
<td>G</td>
<td>A</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
<td>T</td>
<td>C</td>
<td>T</td>
</tr>
<tr>
<td>Sample 4</td>
<td>C</td>
<td>G</td>
<td>A</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>A</td>
<td>C</td>
<td>G</td>
<td>G</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>A</td>
</tr>
</tbody>
</table>

Treat sample 1 as the *reference coordinate system* and identify variants.

A *coordinate system* uniquely identifies the position of a variant in a given genome.
Multiple sample sequences and variants

Treat sample 1 as the **reference coordinate system** and identify variants

* A *coordinate system* uniquely identifies the position of a variant in a given genome
Multiple sample sequences and variants

Treat sample 1 as the **reference coordinate system** and identify variants.

*A coordinate system* uniquely identifies the position of a variant in a given genome.
Multiple sample sequences and variants

Treat sample 1 as the *reference coordinate system* and identify variants.

A *coordinate system* uniquely identifies the position of a variant in a given genome.
Indels introduce multiple coordinate systems

Each sample can have a *different* coordinate system
Variant queries map positions to variants

Reference-only indexes map positions only in the reference coordinate system

\[ f(p_i, p_j) \rightarrow (v_i \ldots v_n), \quad \text{where } p_i \leq p_j \]
Indexing in multiple coordinates is challenging

Reference-only indexes map positions only in the reference coordinate system

\[ f(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \]

Pan-genome analysis involves queries based on sample coordinate systems

\[
\begin{align*}
\{ & f_1(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \\
\vdots & \\
& f_s(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \\
\}
\]

Maintaining thousands of mappings increases computational complexity and memory footprint. **Limits scalability** to population-scale data.
Existing solutions do not scale to thousands of samples

- Existing solutions are built to cater to specific applications
- For example, VG toolkit\(^1\) and Seven Bridges\(^2\) are built for read mapping applications
- They encode variants in a variation graph and perform graph traversals for read mapping
- They support sequence search but do not support other kinds of queries
- The solutions are not designed to scale with increasing amounts of population-level variation data

---

\(^1\) Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, 36:875–879, 2018

Reference-only indexes do not support multiple coordinate queries

- GQT\(^1\), BGT\(^2\), and GTC\(^3\) are *reference-only indexes*
- They are optimized to support positional variant queries but *do not store sequences for comparison*
- Traditional database-based solutions have proven *prohibitively slow*

---

Reference-only indexes do not support multiple coordinate queries

- GQT\textsuperscript{[1]}, BGT\textsuperscript{[2]}, and GTC\textsuperscript{[3]} are reference-only indexes
- They are optimized to support positional variant queries but do not store sequences

Existing systems don’t support multiple coordinate systems. The ones that do, don’t \textit{scale} beyond a few thousand samples.

\[\text{References}\]

\textsuperscript{[3]} GTC: how to maintain huge genotype collections in a compressed form. \textit{Bioinformatics}, 34(11):1834–1840, 2018
VariantStore: a system to efficiently index and query population-level variation data

- Supports querying variants in both reference and *sample-specific coordinates*
  - Takes between 0.002 -- 3 seconds for different types of variant queries
- *Scales* to data containing *thousands of samples* and millions of variants
  - 1000 Genomes project, 2500 samples and 924M variants, 3 Hrs
  - TCGA (BRCA) project, 8640 samples and 5M variants, 4 Hrs
- *Efficiently scales out-of-RAM* to enable memory-efficient construction and query
  - Peak RAM is 10% the size of the index
Variation graph construction

Sample 1: CAATTTGCTGATCT
Variation graph construction

Sample 1: CAATTTGCTGATCT
Sample 2: CATGCTGATCT
Variation graph construction

Sample 1: CAATTTGCTGATCT
Sample 2: CATGCTGATCT
Sample 3: CGATTTGCTGATCT
Variation graph construction

Sample 1: CAATTTGCTGATCT
Sample 2: CATGCTGATCT
Sample 3: CGATTTGCTGATCT
Sample 4: CGATTTACGGCTGATCT
An inverted index on the pan-genome graph

- Succinct index for reference coordinate system
- Local-graph exploration to map position from reference to sample coordinate

Variation graph

Position index
An inverted index on the pan-genome graph

- Partition the variation graph based on coordinate ranges
- Store partitions on disk
- Succinct index for reference coordinate system
- Local-graph exploration to map position from reference to sample coordinate

Queries often require loading 1-2 partitions
VariantStore is $3 \times$ faster, takes $25\%$ less disk space, and $3 \times$ less peak RAM than VG toolkit.

### Results for constructing the index

<table>
<thead>
<tr>
<th>System</th>
<th>Time</th>
<th>Disk space</th>
<th>Peak RAM</th>
<th>Peak RAM Agg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td></td>
<td>1000 Genomes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VariantStore</td>
<td>3 Hrs 25 mins</td>
<td>41 GB</td>
<td>8.8 GB</td>
<td>153 GB</td>
</tr>
<tr>
<td>VG-toolkit</td>
<td>11 Hrs 10 mins</td>
<td>50 GB</td>
<td>37 GB</td>
<td>450 GB</td>
</tr>
<tr>
<td>Dataset</td>
<td>TCGA (OV)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VariantStore</td>
<td>1 Hr 5 mins</td>
<td>3.4 GB</td>
<td>1.1 GB</td>
<td>17.45 GB</td>
</tr>
<tr>
<td>VG-toolkit</td>
<td></td>
<td>11 GB*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset</td>
<td>TCGA (LUAD)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VariantStore</td>
<td>1 Hr 20 mins</td>
<td>3.5 GB</td>
<td>2.3 GB</td>
<td>36.05 GB</td>
</tr>
<tr>
<td>VG-toolkit</td>
<td></td>
<td>12 GB*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset</td>
<td>TCGA (BRCA)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VariantStore</td>
<td>4 Hrs 36 mins</td>
<td>4.2 GB</td>
<td>3.2 GB</td>
<td>53.21 GB</td>
</tr>
<tr>
<td>VG-toolkit</td>
<td></td>
<td>14 GB*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Time, space, peak RAM, and peak RAM (aggregate) to construct variant index on the 1000 Genomes and TCGA (OV, LUAD, and BRCA) data using VariantStore and VG toolkit. *VG toolkit could not build GBWT index embedding all sample paths for TCGA data. Space reported is for the XG index that does not contain any path information. We constructed all 24 chromosomes (1 – 22 and X and Y) in parallel. The time and peak RAM reported is for the biggest chromosome (usually chromosome 1 or 2). The space reported is the total space on disk for all 24 chromosomes. The peak RAM (aggregate) is the aggregate peak RAM for all 24 processes.
Aggregate time to execute queries *increases sublinearly* with the number of queries
Query analysis based on range size

Memory usage *remains constant* regardless of the query length.
The ability to efficiently query population-scale variation data promises to improve multiple downstream applications.

VariantStore enables variant queries across thousands of samples.

VariantStore efficiently scales out of RAM for easy usage in limited memory environments.

https://github.com/Kingsford-Group/variantstore
Acknowledgements

Carnegie Mellon University
Carl Kingsford
Guillaume Marcais
Dan DeBlasio

Berkeley Lab/UC Berkeley
Kathy Yelick
Aydin Buluc

Stony Brook University
Rob Patro
Rob Johnson
Michael Bender
Fatemeh Almodaresi

https://prashantpandey.github.io
The VCF format has been developed to encode variants from large scale sequencing. These files contain variation as mutations based on a reference genome:
- SNPs and Indels
Applications performing pan-genome variant queries

- To determine the region of interest in PCR primer design [1], applications extract a fixed length sequence up and downstream from a given variant location in samples that share the variant and look for nearby variants affecting the primer.

- In colocalization analysis [2], applications query and compare variants or sequences in a genomic region across samples to determine significant overlaps between genomic regions in order to establish evolutionary or mechanistic relationships.
