New models and algorithms for genome comparison

Jens Stoye

Genome Informatics, Faculty of Technology
and
Institute of Bioinformatics, Center of Biotechnology

Bielefeld University, Germany
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Overview: New models and algorithms for genome comparison

• Introduction
  – Comparative genomics
  – Gene clusters

• Finding gene clusters
  – Gene clusters of permutations
  – Gene clusters of sequences
  – Experimental results

• Conserved intervals
  – Finding conserved intervals
  – Application to mitochondrial genomes

• Summary and Conclusion
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Overview: Completely sequenced genomes

(from GOLD database: http://wit.integratedgenomics.com/GOLD/)

182 published complete genomes (including 4 chromosomes):

- **141 bacterial genomes** (first: *H. influenzae*, 1995)
  - size: \(\approx 500 \ldots 10,000\) kilobases (KB)
  - genes: \(\approx 450 \ldots 10,000\) open reading frames (ORFs)

- **18 archaeal genomes** (first: *M. janaschii*, 1996)
  - size: \(\approx 1,500 \ldots 6,000\) KB
  - genes: \(\approx 1,500 \ldots 4,500\) ORFs

- **23 eukaryal genomes** (first: *S. cerevisiae*, 1997)
  - size: \(\approx 12,000 \ldots 2,800,000\) KB
  - genes: \(\approx 6,000 \ldots 50,000\) ORFs
Overview: Completely sequenced genomes
(from GOLD database: http://wit.integratedgenomics.com/GOLD/)

182 published complete genomes (including 4 chromosomes):

• **141 bacterial genomes** (first: *H. influenzae*, 1995)
  – size: \( \approx 500 \ldots 10,000 \) kilobases (KB)
  – genes: \( \approx 450 \ldots 10,000 \) open reading frames (ORFs)

• **18 archaeal genomes** (first: *M. janaschii*, 1996)
  – size: \( \approx 1,500 \ldots 6,000 \) KB
  – genes: \( \approx 1,500 \ldots 4,500 \) ORFs

• **23 eukaryal genomes** (first: *S. cerevisiae*, 1997)
  – size: \( \approx 12,000 \ldots 2,800,000 \) KB
  – genes: \( \approx 6,000 \ldots 50,000 \) ORFs

Next steps:
  functional genomics (transcriptomics, proteomics, metabolomics, ...)
  comparative genomics
Comparative genomics “at a higher level”

Concentrate on large scale layout of the genomes:

- Study genomes based on their *gene order*.
- Represent genomes by their sequence of genes.

```
genome 1

---

genome 2
```
Comparative genomics “at a higher level”

Concentrate on large scale layout of the genomes:

- Study genomes based on their gene order.
- Represent genomes by their sequence of genes.

\[\text{genome 1}\]
\[\text{genome 2}\]
Comparative genomics “at a higher level”

Concentrate on large scale layout of the genomes:

• Study genomes based on their *gene order*.
• Represent genomes by their sequence of genes.

![Graph showing genome order](image)

More formally:

• Genes = (signed) elements from the set $N = \{0, \ldots, n\}$.
• Assign the same number to corresponding (*orthologous*) genes.
• Genomes = permutations of $N$. 
Genome rearrangement operations and distances

→ rearrangement operations: (signed) reversal

\[ +0 +1 +2 +3 +4 +5 \rightarrow +0 -4 -3 -2 -1 +5 \]

transposition

\[ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \rightarrow 0 \ 3 \ 4 \ 1 \ 2 \ 5 \]

Resulting distances and problems:

• (signed) reversal distance → sorting (signed) permutations by reversals

• transposition distance → sorting permutations by transpositions

Generalization: multiple chromosomes

→ additional operations: fission

\[ \text{fission} \]

\[ \rightarrow \]

fusion

\[ \rightarrow \]

translocation

\[ \rightarrow \]

If gene order is unknown: syntenic distance (chromosomes as bags of genes)
Sorting by reversals

**Problem:** Given two (signed) permutations (genomes) $\pi_1$ and $\pi_2$ of the elements (genes) of the set $\mathbb{N} = \{0, 1, \ldots, n\}$, find the minimal number of *reversals* that are necessary to transform $\pi_1$ into $\pi_2$.

\[
\begin{array}{ccccccc}
\pi_1 & +4 & +5 & +2 & +3 & +0 & +1 \\
\pi_2 & +0 & +1 & +2 & +3 & +4 & +5 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
+4 & +5 & +2 & +3 & +0 & +1 \\
-3 & -2 & -5 & -4 & +0 & +1 \\
-3 & -2 & -1 & -0 & +4 & +5 \\
+0 & +1 & +2 & +3 & +4 & +5 \\
\end{array}
\]

**Similar:** *transposition distance, translocation distance, ...*
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Protein function prediction

About 30% of the ORFs in the MIPS Yeast Genome Database still have no function annotation.

Functional annotation is time consuming and expensive (≈ 1 – 2 years, ≈ 1 – 2 million US$ per gene).
Protein function prediction

About 30% of the ORFs in the MIPS Yeast Genome Database still have no function annotation.

Functional annotation is time consuming and expensive (≈ 1 – 2 years, ≈ 1 – 2 million US$ per gene).

- In the lab:
  - Genetical and biochemical analysis
  - Correlated expression
Protein function prediction

About 30% of the ORFs in the MIPS Yeast Genome Database still have no function annotation.

Functional annotation is time consuming and expensive ($\approx 1 - 2$ years, $\approx 1 - 2$ million US$ per gene).

- In the lab:
  - Genetical and biochemical analysis
  - Correlated expression

- Homology based:
  - Protein families
  - Functional domains
Protein function prediction

About 30% of the ORFs in the MIPS Yeast Genome Database still have no function annotation.

Functional annotation is time consuming and expensive ($\approx 1 – 2$ years, $\approx 1 – 2$ million US$ per gene).

- In the lab:
  - Genetical and biochemical analysis
  - Correlated expression
- Homology based:
  - Protein families
  - Functional domains
- Genome based:
  - Rosetta stone method (gene fusion, domain fusion)
  - Phylogenetic profiles (correlated evolution)
  - Gene order (co-occurrence of genes in genomes)
Protein function prediction

About 30% of the ORFs in the MIPS Yeast Genome Database still have no function annotation.

Functional annotation is time consuming and expensive (≈ 1 – 2 years, ≈ 1 – 2 million US$ per gene).

- In the lab:
  - Genetical and biochemical analysis
  - Correlated expression
- Homology based:
  - Protein families
  - Functional domains
- Genome based:
  - Rosetta stone method (gene fusion, domain fusion)
  - Phylogenetic profiles (correlated evolution)
  - Gene order (co-occurrence of genes in genomes)
- Literature based:
  - Natural language processing
Genome-based gene function prediction

Functional genomics meets comparative genomics.

Idea: Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway
Genome-based gene function prediction

*Functional genomics meets comparative genomics.*

**Idea:** Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway
**Genome-based gene function prediction**

*Functional genomics meets comparative genomics.*

**Idea:** Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway
Genome-based gene function prediction

*Functional genomics meets comparative genomics.*

**Idea:** Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway
Genome-based gene function prediction

*Functional genomics meets comparative genomics.*

**Idea:** Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

– interacting proteins
– proteins of the same protein complex
– enzymes of the same metabolic pathway
Genome-based gene function prediction

*Functional genomics meets comparative genomics.*

**Idea:** Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway
Genome-based gene function prediction

*Functional genomics meets comparative genomics.*

**Idea:** Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway
Genome-based gene function prediction

Functional genomics meets comparative genomics.

Idea: Genes that repeatedly cluster together in phylogenetically remotely related genomes are functionally associated:

- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway


STRING Web server (Snel et al., 2000)

http://string.embl.de/
STRING Web server (Snel et al., 2000)

http://string.embl.de/
Genome Windows: DCW cluster (division and cell wall)

Jens Stoye: New models and algorithms for genome comparison
Genome Windows: Ribose-ABC-Transporter

Genome Windows of Cluster 79903:

- **Bacillus subtilis**

- **Escherichia coli K-12 MG1655**

- **Escherichia coli UT5717**

- **Hemophilus influenzae**

- **Thermotoga maritima**

Jens Stoye: *New models and algorithms for genome comparison*
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur contiguously in all permutations.

\[
\begin{array}{cccccccc}
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & 7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}
\]
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur contiguously in all permutations.

$$\begin{array}{cccccccc}
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & 7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}$$

Common intervals: $[3,4]$
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur contiguously in all permutations.

\[
\begin{array}{cccccccc}
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & 7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}
\]

Common intervals: [3,4] [2,4]
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) 0, 1, \ldots, $n$, find subsets of numbers that occur contiguously in all permutations.

$$
\begin{array}{c|cccccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & 7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}
$$

Common intervals: [3,4] [2,4] [1,4]
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur contiguously in all permutations.

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>0 1 2 3 4 5 6 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_2$</td>
<td>6 7 5 1 4 3 2 0</td>
</tr>
<tr>
<td>$\pi_3$</td>
<td>7 0 1 2 4 3 5 6</td>
</tr>
</tbody>
</table>

Common intervals: [3,4] [2,4] [1,4] [0,4]
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur **contiguously** in all permutations.

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>0 1 2 3 4 5 6 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_2$</td>
<td>6 7 5 1 4 3 2 0</td>
</tr>
<tr>
<td>$\pi_3$</td>
<td>7 0 1 2 4 3 5 6</td>
</tr>
</tbody>
</table>

Common intervals: [3,4] [2,4] [1,4] [0,4] [1,5]
Formalization of gene cluster: **common interval**

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) 0, 1, $\ldots$, $n$, find subsets of numbers that occur *contiguously* in all permutations.

$\pi_1$  
0 1 2 3 4 5 6 7

$\pi_2$  
6 7 5 1 4 3 2 0

$\pi_3$  
7 0 1 2 4 3 5 6

Common intervals: [3,4] [2,4] [1,4] [0,4] [1,5] [0,5]
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur contiguously in all permutations.

\[
\begin{array}{cccccccc}
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & 7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}
\]

Common intervals: $[3,4] \ [2,4] \ [1,4] \ [0,4] \ [1,5] \ [0,5] \ [0,7]$
Formalization of gene cluster: common interval

Given permutations (genomes) $\pi_1, \pi_2, \ldots, \pi_k$ of the numbers (genes) $0, 1, \ldots, n$, find subsets of numbers that occur contiguously in all permutations.

$$
\begin{array}{cccccccc}
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & 7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}
$$

Common intervals: [3,4] [2,4] [1,4] [0,4] [1,5] [0,5] [0,7]

Algorithms:

- Uno & Yagiura, *Algorithmica* 2000:
  Find all common intervals of 2 permutations in $O(n + |\text{output}|)$ time.

- Heber & Stoye, *CPM* 2001:
  Find all common intervals of $k \geq 2$ permutations in $O(kn + |\text{output}|)$ time.
Finding all common intervals of 2 permutations $\pi_1$ and $\pi_2$

Let $1 \leq x \leq y \leq n$.

Notation: $\pi([x, y]) := \{\pi(x), \pi(x + 1), \ldots, \pi(y)\}$

Definitions: $l(x, y) := \min \pi_2([x, y])$
$u(x, y) := \max \pi_2([x, y])$
$f(x, y) := u(x, y) - l(x, y) - (y - x)$

Example:

\[
\begin{array}{cccccccc}
\pi_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]
Finding all common intervals of 2 permutations $\pi_1$ and $\pi_2$

Let $1 \leq x \leq y \leq n$.

**Notation:**

$$\pi([x, y]) := \{\pi(x), \pi(x + 1), \ldots, \pi(y)\}$$

**Definitions:**

$$l(x, y) := \min_{\pi_2} [x, y]$$

$$u(x, y) := \max_{\pi_2} [x, y]$$

$$f(x, y) := u(x, y) - l(x, y) - (y - x)$$

**Example:**

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

$$f(1, 4) = 4 - 1 - (6 - 3) = 0$$
Finding all common intervals of 2 permutations $\pi_1$ and $\pi_2$

Let $1 \leq x \leq y \leq n$.

Notation: $\pi([x, y]) := \{\pi(x), \pi(x + 1), \ldots, \pi(y)\}$

Definitions: $l(x, y) := \min \pi_2([x, y])$
               $u(x, y) := \max \pi_2([x, y])$
               $f(x, y) := u(x, y) - l(x, y) - (y - x)$

Example:

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

$f(1, 4) = 4 - 1 - (6 - 3) = 0$

<table>
<thead>
<tr>
<th>$\pi_2$</th>
<th>6</th>
<th>7</th>
<th>5</th>
<th>1</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

$f(1, 4) = 7 - 1 - (4 - 1) = 3 > 0$
Finding all common intervals of 2 permutations $\pi_1$ and $\pi_2$

Let $1 \leq x \leq y \leq n$.

Notation: $\pi([x, y]) := \{\pi(x), \pi(x + 1), \ldots, \pi(y)\}$

Definitions: $l(x, y) := \min \pi_2([x, y])$
             $u(x, y) := \max \pi_2([x, y])$
             $f(x, y) := u(x, y) - l(x, y) - (y - x)$

Example:

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

$\pi_2$

<table>
<thead>
<tr>
<th>6</th>
<th>7</th>
<th>5</th>
<th>1</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

$f(1, 4) = 4 - 1 - (6 - 3) = 0$

$f(1, 4) = 7 - 1 - (4 - 1) = 3 > 0$

Simple algorithm: For all $1 \leq x \leq y \leq n$ test if $f(x, y) = 0$.

Analysis: $\mathcal{O}(n^2)$ time.
Finding all common intervals of two permutations $\pi_1$ and $\pi_2$

Uno & Yagiura, 2000:
Perform the test $f(x, y) = 0$ not for all pairs $(x, y)$.

Definition:
For given $x$, call a value of $y > x$ wasteful, if and only if for all $x' \leq x$:

$$f(x', y) > 0.$$ 

Lemma:
For fixed $x$, $f(x, y)$ increases monotonically for the non-wasteful indices $y (> x)$.

Algorithm (Idea):
• $x$ runs in right-to-left direction through a doubly linked list $ylist$ that initially contains the entries of $\pi_2$.
• In each step, the entries of wasteful indices $y (> x)$ are removed.
• Test for the remaining $y > x$ in $ylist$ from left to right if $f(x, y) = 0$. 

Jens Stoye: New models and algorithms for genome comparison
Algorithm RC (Uno & Yagiura)

- Removal of wasteful indices from $ylist$ is done by means of two additional lists $llist$ and $ulist$ that implement the functions $l$ and $u$.

- The elements of $llist$ and $ulist$ are maximal intervals of $ylist$ with the same smallest resp. largest element.

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
| ylist | 6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 |
| llist | 6 & 5 & 1 & 0 |
| ulist | 6 & 7 |
\end{array}
\]
Algorithm RC (Uno & Yagiura)

- **Removal of wasteful indices** from \( ylist \) is done by means of two additional lists \( llist \) and \( ulist \) that implement the functions \( l \) and \( u \).

- The elements of \( llist \) and \( ulist \) are maximal intervals of \( ylist \) with the same smallest resp. largest element.

Analysis:
\( \mathcal{O}(n + |output|) \) time, \( \mathcal{O}(n) \) space.
Finding all common intervals of \( k \geq 2 \) permutations

**Obvious generalization:**
Given \( k \) permutations \( \pi_1, \pi_2, \ldots, \pi_k \).
For \( j = 2, 3, \ldots, k \) compute the common intervals of \( \pi_1 \) and \( \pi_j \).
Output all intervals that are found in all of these comparisons.

\[
\begin{array}{ccccccccc}
\pi_1 & & & & & & & & \\
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\pi_2 & & & & & & & & \\
6 & 7 & 5 & 1 & 4 & 3 & 2 & 0 \\
\pi_3 & & & & & & & & \\
7 & 0 & 1 & 2 & 4 & 3 & 5 & 6 \\
\end{array}
\]
Finding all common intervals of $k \geq 2$ permutations

Obvious generalization:
Given $k$ permutations $\pi_1, \pi_2, \ldots, \pi_k$.
For $j = 2, 3, \ldots, k$ compute the common intervals of $\pi_1$ and $\pi_j$.
Output all intervals that are found in all of these comparisons.

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>0 1 2 3 4 5 6 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_2$</td>
<td>6 7 5 1 4 3 2 0</td>
</tr>
<tr>
<td>$\pi_3$</td>
<td>7 0 1 2 4 3 5 6</td>
</tr>
</tbody>
</table>

[0,4] [0,5] [0,7] [1,4] [1,5] [1,7] [2,3] [2,4] [3,4] [5,7] [6,7]
Finding all common intervals of $k \geq 2$ permutations

Obvious generalization:
Given $k$ permutations $\pi_1, \pi_2, \ldots, \pi_k$.
For $j = 2, 3, \ldots, k$ compute the common intervals of $\pi_1$ and $\pi_j$.
Output all intervals that are found in all of these comparisons.
Finding all common intervals of $k \geq 2$ permutations

**Obvious generalization:**
Given $k$ permutations $\pi_1, \pi_2, \ldots, \pi_k$.
For $j = 2, 3, \ldots, k$ compute the common intervals of $\pi_1$ and $\pi_j$.
Output all intervals that are found in all of these comparisons.
Finding all common intervals of $k \geq 2$ permutations

Obvious generalization:
Given $k$ permutations $\pi_1, \pi_2, \ldots, \pi_k$.
For $j = 2, 3, \ldots, k$ compute the common intervals of $\pi_1$ and $\pi_j$.
Output all intervals that are found in all of these comparisons.

Analysis:
$O(kn + \sum |K_i|)$ time
where $K_i =$ the number of common intervals of $\pi_1$ and $\pi_i$. 
Irreducible Intervals

**Goal:** An algorithm with output-dependent time complexity $O(kn + |\text{output}|)$.

**Observation:** Common intervals form “chains” of non-trivially overlapping intervals.

![Diagram of intervals and their corresponding chains](image)

**Definition:**
A common interval $c$ is **reducible** if there exists a non-trivial chain that generates $c$, otherwise it is **irreducible**.
Properties of irreducible intervals

Lemma:
The subchains of all the maximal chains of irreducible intervals generate exactly all common intervals.

Theorem: For is the number of irreducible intervals $K$ the following holds:

$$1 \leq K \leq n - 1$$

Example:

$K = 1$

$K = n - 1$
Finding all common intervals of $k \geq 2$ permutations

Algorithm:

- Find the set of all irreducible intervals.
- Partition this set into maximal chains of non-trivially overlapping intervals.
- For each such chain generate all subchains: the common intervals.
Finding all common intervals of $k \geq 2$ permutations

Algorithm:

- Find the set of all irreducible intervals.
- Partition this set into maximal chains of non-trivially overlapping intervals.
- For each such chain generate all subchains: the common intervals.
Finding all common intervals of $k \geq 2$ permutations

Algorithm:

- Find the set of all irreducible intervals.
- Partition this set into maximal chains of non-trivially overlapping intervals.
- For each such chain generate all subchains: the common intervals.

Analysis: $\mathcal{O}(kn + |\text{output}|)$ time, $\mathcal{O}(n)$ additional space
More realistic genome models

1. Genomes of higher organisms often have more than one chromosome ⇒ multichromosomal permutations

\[
\begin{align*}
\pi_1 & : 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\pi_2 & : 5 & 1 & 0 & 2 & 3 & 4 & 6 & 8 & 7 \\
\end{align*}
\]

2. Genes of a cluster should lie on the same DNA strand ⇒ signed permutations

\[
\begin{align*}
\pi_1 & : +0 & +1 & +2 & +3 & +4 & +5 & +6 & +7 & +8 \\
\pi_2 & : +8 & +7 & +3 & +5 & +4 & -6 & -0 & -1 & -2 \\
\end{align*}
\]

3. Bacterial, archaeal, and mitochondrial DNA is often circular ⇒ circular permutations

\[
\begin{align*}
\pi_1 & : 7 & 8 & 0 & 1 & 6 & 5 & 4 & 3 & 2 \\
\pi_2 & : 8 & 1 & 0 & 3 & 6 & 5 & 4 & 2 & 7 \\
\end{align*}
\]
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Inclusion of paralogous genes

Problem:
In case of duplicated genes, it is difficult to assign correct orthologous gene pairs. Possibly the ortholog does not even exist.

Consequence:
Do not distinguish between paralogous gene copies.

New model:
Use the same element (number) more than once for paralogous copies of genes. → genomes are modeled as sequences instead of permutations.
Formal model

Given: $k$ sequences $S = (S_1, S_2, \ldots, S_k)$ over an alphabet $\Sigma$.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_i \in S$.

Goal:
Find all maximal occurrences of common intervals in $S$. 

Jens Stoye: New models and algorithms for genome comparison 28
Formal model

**Given:** $k$ sequences $S = (S_1, S_2, \ldots, S_k)$ over an alphabet $\Sigma$.

**Common interval:**
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_i \in S$.

**Goal:**
Find all maximal occurrences of common intervals in $S$.

**Example:**

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 1 2 3 1 5 2 6</td>
<td>4 3 5 5 5 1 4 2 2</td>
<td>7 5 1 5 3 6 5</td>
</tr>
</tbody>
</table>
 Formal model

Given: $k$ sequences $S = (S_1, S_2, \ldots, S_k)$ over an alphabet $\Sigma$.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_i \in S$.

Goal:
Find all maximal occurrences of common intervals in $S$.

Example:

\[
\begin{array}{cccccccc}
S_1 & 3 & 1 & 2 & 3 & 1 & 5 & 2 & 6 \\
S_2 & 4 & 3 & 5 & 5 & 5 & 1 & 4 & 2 & 2 \\
S_3 & 7 & 5 & 1 & 5 & 3 & 6 & 5 \\
\end{array}
\]

Common intervals: $\{3\}$
Formal model

**Given:** $k$ sequences $S = (S_1, S_2, \ldots, S_k)$ over an alphabet $\Sigma$.

**Common interval:**
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_i \in S$.

**Goal:**
Find all maximal occurrences of common intervals in $S$.

**Example:**

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 1 2 3 1 5 2 6</td>
<td>4 3 5 5 5 1 4 2 2</td>
<td>7 5 1 5 3 6 5</td>
</tr>
</tbody>
</table>

Common intervals: $\{3\} \quad \{1\}$
Formal model

Given: $k$ sequences $S = (S_1, S_2, \ldots, S_k)$ over an alphabet $\Sigma$.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_i \in S$.

Goal:
Find all maximal occurrences of common intervals in $S$.

Example:

$$
\begin{array}{cccccccc}
S_1 & 3 & 1 & 2 & 3 & 1 & 5 & 2 & 6 \\
S_2 & 4 & 3 & 5 & 5 & 5 & 1 & 4 & 2 & 2 \\
S_3 & 7 & 5 & 1 & 5 & 3 & 6 & 5 \\
\end{array}
$$

Common intervals: \{3\} \{1\} \{5\}
Formal model

Given: \( k \) sequences \( S = (S_1, S_2, \ldots, S_k) \) over an alphabet \( \Sigma \).

Common interval:
a subset \( C \subseteq \Sigma \) whose elements occur contiguously in each \( S_i \in S \).

Goal:
Find all maximal occurrences of common intervals in \( S \).

Example:

\[
\begin{array}{ccccccccc}
S_1 & 3 & 1 & 2 & 3 & 1 & 5 & 2 & 6 \\
S_2 & 4 & 3 & 5 & 5 & 5 & 1 & 4 & 2 & 2 \\
S_3 & 7 & 5 & 1 & 5 & 3 & 6 & 5 \\
\end{array}
\]

Common intervals: \( \{3\} \quad \{1\} \quad \{5\} \quad \{1,5\} \)
Formal model

Given: \( k \) sequences \( S = (S_1, S_2, \ldots, S_k) \) over an alphabet \( \Sigma \).

Common interval:
a subset \( C \subseteq \Sigma \) whose elements occur contiguously in each \( S_i \in S \).

Goal:
Find all maximal occurrences of common intervals in \( S \).

Example:

\[
\begin{align*}
S_1 &:\quad 3 & 1 & 2 & 3 & 1 & 5 & 2 & 6 \\
S_2 &:\quad 4 & 3 & 5 & 5 & 5 & 1 & 4 & 2 & 2 \\
S_3 &:\quad 7 & 5 & 1 & 5 & 3 & 6 & 5 \\
\end{align*}
\]

Common intervals: \( \{3\} \quad \{1\} \quad \{5\} \quad \{1,5\} \quad \{1,3,5\} \)
Formal model

Given: $k$ sequences $S = (S_1, S_2, \ldots, S_k)$ over an alphabet $\Sigma$.

Common interval: 
A subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_i \in S$.

Goal: 
Find all maximal occurrences of common intervals in $S$.

Example:

\[
\begin{array}{cccccccc}
  S_1 & 3 & 1 & 2 & 3 & 1 & 5 & 2 & 6 \\
  S_2 & 4 & 3 & 5 & 5 & 5 & 1 & 4 & 2 & 2 \\
  S_3 & 7 & 5 & 1 & 5 & 3 & 6 & 5 \\
\end{array}
\]

Common intervals: \{3\} \{1\} \{5\} \{1,5\} \{1,3,5\}
An elementary algorithm for two sequences

Preprocessing: compute two tables for $S_1 = (3, 1, 2, 3, 1, 5, 2, 6)$:

| $POS[1]$ | 2, 5 |
| $POS[2]$ | 3, 7 |
| $POS[3]$ | 1, 4 |
| $POS[4]$ | empty |
| $POS[5]$ | 6 |
| $POS[6]$ | 8 |

$NUM(i, j):$

<table>
<thead>
<tr>
<th>$i \backslash j$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Algorithm:

While reading $S_2$, mark in $S_1$ the observed characters and track maximal intervals of marked characters.

$S_1$

<table>
<thead>
<tr>
<th>3</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>5</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
</table>

$S_2$

| 4 | 3 | 5 | 5 | 5 | 1 | 4 | 2 | 2 |
An elementary algorithm for two sequences

Preprocessing: compute two tables for $S_1 = (3, 1, 2, 3, 1, 5, 2, 6)$:

<table>
<thead>
<tr>
<th>$POS[i]$</th>
<th>2, 5</th>
<th>3, 7</th>
<th>1, 4</th>
<th>empty</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM($i, j$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Algorithm:
While reading $S_2$, mark in $S_1$ the observed characters and track maximal intervals of marked characters.
An elementary algorithm for two sequences

Preprocessing: compute two tables for \( S_1 = (3, 1, 2, 3, 1, 5, 2, 6) \):

\[
\begin{align*}
POS[1] &= 2, 5 \\
POS[2] &= 3, 7 \\
POS[3] &= 1, 4 \\
POS[4] &= \text{empty} \\
POS[5] &= 6 \\
POS[6] &= 8
\end{align*}
\]

\[
\begin{array}{ccccccccc}
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 5 \\
1 & 1 & 2 & 3 & 3 & 4 & 4 & 5 & \ \\
2 & 1 & 2 & 3 & 4 & 4 & 5 & & \\
3 & 1 & 2 & 3 & 4 & 5 & & & \\
4 & & & & 1 & 2 & 3 & 4 & \\
5 & & & & 1 & 2 & 3 & & \\
6 & & & & & & 1 & 2 & \\
7 & & & & & & & & 1 \\
\end{array}
\]

Algorithm:
While reading \( S_2 \), mark in \( S_1 \) the observed characters and track maximal intervals of marked characters.
An elementary algorithm for two sequences

Preprocessing: compute two tables for \( S_1 = (3, 1, 2, 3, 1, 5, 2, 6) \):

\[
\begin{align*}
POS[1] &= 2, 5 \\
POS[2] &= 3, 7 \\
POS[3] &= 1, 4 \\
POS[4] &= \text{empty} \\
POS[5] &= 6 \\
POS[6] &= 8 \\
\end{align*}
\]

\[
\begin{array}{c|cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 0 & & & 3 & 3 & 4 & 4 & 5 \\
1 & & & & & & & & \\
2 & 1 & 2 & 3 & 4 & 4 & 5 \\
3 & 1 & 2 & 3 & 4 & 5 & & \\
4 & & & & & & & & \\
5 & & & & & & & & \\
6 & & & & & & & & \\
7 & & & & & & & & \\
\end{array}
\]

Algorithm:
While reading \( S_2 \), mark in \( S_1 \) the observed characters and track maximal intervals of marked characters.

\[
\begin{align*}
S_1 &= [3, 1, 2, 3, 1, 5, 2, 6] \\
S_2 &= [4, 3, 5, 5, 5, 1, 4, 2, 2] \\
\end{align*}
\]
An elementary algorithm for two sequences

Preprocessing: compute two tables for $S_1 = (3, 1, 2, 3, 1, 5, 2, 6)$:

\[
\begin{align*}
POS[1] &= 2, 5 \\
POS[2] &= 3, 7 \\
POS[3] &= 1, 4 \\
POS[4] &= \text{empty} \\
POS[5] &= 6 \\
POS[6] &= 8 \\
\end{align*}
\]

NUM($i, j$) :

\[
\begin{array}{cccccccc}
  & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 5 \\
1 & 1 & 2 & 3 & 3 & 4 & 4 & 4 & 5 \\
2 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 \\
3 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\
4 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 \\
5 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 \\
6 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\
7 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\
\end{array}
\]

Algorithm:
While reading $S_2$, mark in $S_1$ the observed characters and track maximal intervals of marked characters.

$S_1 \quad \boxed{3 \ 1 \ 2 \ 3 \ 1 \ 5 \ 2 \ 6} \quad S_2 \quad \boxed{4 \ 3 \ 5 \ 5 \ 5 \ 1 \ 4 \ 2 \ 2}$
An elementary algorithm for two sequences

Preprocessing: compute two tables for \( S_1 = (3, 1, 2, 3, 1, 5, 2, 6) \):

\[
\begin{align*}
POS[1] &= 2, 5 \\
POS[2] &= 3, 7 \\
POS[3] &= 1, 4 \\
POS[4] &= \text{empty} \\
POS[5] &= 6 \\
POS[6] &= 8 \\
\end{align*}
\]

\[
NUM(i, j) : \begin{array}{cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 5 \\
1 & 1 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \\
2 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 \\
3 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\
4 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 \\
5 & 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 \\
6 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\
7 & 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\
\end{array}
\]

Algorithm:
While reading \( S_2 \), mark in \( S_1 \) the observed characters and track maximal intervals of marked characters.
An elementary algorithm for two sequences

Preprocessing: compute two tables for $S_1 = (3, 1, 2, 3, 1, 5, 2, 6)$:

$\begin{align*}
\text{POS}[1] &= 2, 5 \\
\text{POS}[2] &= 3, 7 \\
\text{POS}[3] &= 1, 4 \\
\text{POS}[4] &= \text{empty} \\
\text{POS}[5] &= 6 \\
\text{POS}[6] &= 8
\end{align*}$

$\text{NUM}(i, j):$

$\begin{array}{ccccccccc}
\text{i} & \backslash \text{j} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 1 & 2 & 3 & 3 & 3 & 4 & 4 & 5 \\
1 & 1 & 2 & 3 & 3 & 4 & 4 & 5 \\
2 & 1 & 2 & 3 & 4 & 4 & 5 \\
3 & & 1 & 2 & 3 & 4 & 5 \\
4 & & & 1 & 2 & 3 & 4 \\
5 & & & & 1 & 2 & 3 \\
6 & & & & & 1 & 2 \\
7 & & & & & & 1
\end{array}$

Algorithm:
While reading $S_2$, mark in $S_1$ the observed characters and track maximal intervals of marked characters.

$\begin{array}{cccccccc}
\text{S}_1 & 3 & 1 & 2 & 3 & 1 & 5 & 2 & 6 \\
\text{S}_2 & 4 & 3 & 5 & 5 & 5 & 1 & 4 & 2 & 2
\end{array}$

Analysis: $O(n^2)$ time and space.
More algorithms

Space reduction:

- A different algorithm based on work by Didier (CPM, 2003) finds all common intervals of two sequences in $O(n^2)$ time and $O(n)$ space.

More than two sequences:

- Find all common intervals in $k$ sequences in $O(kn^2)$ time and space.

- Find all common intervals that appear in at least $k'$ out of $k$ given sequences in $O(k(1 + k - k')n^2)$ time and $O(kn^2)$ space.
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Experimental results. Data source: COG

Aquifex aeolicus complete genome - 0..1551335
1529 proteins

<table>
<thead>
<tr>
<th>Location</th>
<th>Strand</th>
<th>Length</th>
<th>PID</th>
<th>Synonym</th>
<th>Code</th>
<th>COG</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1..2100</td>
<td>+</td>
<td>699</td>
<td>15605613</td>
<td>fusA</td>
<td>J</td>
<td>COG0480</td>
<td>elongation factor EF-G</td>
</tr>
<tr>
<td>17..3334</td>
<td>+</td>
<td>405</td>
<td>15605614</td>
<td>tufA1</td>
<td>J</td>
<td>COG0050</td>
<td>elongation factor EF-Tu</td>
</tr>
<tr>
<td>46..3660</td>
<td>+</td>
<td>104</td>
<td>15605615</td>
<td>rpsJ</td>
<td>J</td>
<td>COG0051</td>
<td>ribosomal protein S10</td>
</tr>
<tr>
<td>3665..4390</td>
<td>+</td>
<td>241</td>
<td>15605616</td>
<td>rplC</td>
<td>J</td>
<td>COG0087</td>
<td>ribosomal protein L03</td>
</tr>
<tr>
<td>4387..4986</td>
<td>+</td>
<td>199</td>
<td>15605617</td>
<td>rplD</td>
<td>J</td>
<td>COG0088</td>
<td>ribosomal protein L04</td>
</tr>
<tr>
<td>4990..5301</td>
<td>+</td>
<td>103</td>
<td>15605618</td>
<td>rplW</td>
<td>J</td>
<td>COG0089</td>
<td>ribosomal protein L23</td>
</tr>
<tr>
<td>5313..6227</td>
<td>+</td>
<td>304</td>
<td>15605619</td>
<td>rplB</td>
<td>J</td>
<td>COG0090</td>
<td>ribosomal protein L02</td>
</tr>
<tr>
<td>6340..6900</td>
<td>+</td>
<td>186</td>
<td>15605620</td>
<td>rpsS</td>
<td>J</td>
<td>COG0185</td>
<td>ribosomal protein S19</td>
</tr>
<tr>
<td>7018..7314</td>
<td>+</td>
<td>98</td>
<td>15605621</td>
<td>rplV</td>
<td>J</td>
<td>COG0091</td>
<td>ribosomal protein L22</td>
</tr>
</tbody>
</table>

480 50 51 87 88 89
Experimental results. Application to 43 bacterial genomes

- Without closely related genomes (k=32):
  - Cluster size ≥ 2 (2734 clusters)
- All genomes (k=43):
  - Cluster size ≥ 2 (3768 clusters)

- Cluster size ≥ 3:
  - 999 clusters
  - 2043 clusters
Experimental results. Graphical inspection of gene clusters
Experimental results. Graphical inspection of gene clusters
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
On genomic distances

So far: use gene clusters for functional genomics

More traditional approach in genome rearrangement studies:
use gene order data to estimate evolutionary divergence of genomes.

Definition: The **XXX distance** between two permutations is the **minimum**
number of **XXX** operations that transform one permutation into the other.

1998; Christie 1998; Kaplan, Shamir & Tarjan 1999; Bader, Moret & Yan 2001;
Bergeron 2001; Siepel 2002.

Alternate approach: Find **structures** that are shared by two permutations that
are **invariant** under optimal, or biologically meaningful, rearrangement scenarios.

History (partial): Blanchette, Kunisawa & Sankoff 1999; Uno & Yagiura 2000;
First approach: adjacencies/breakpoints

A pair of genes \((a, b)\) is a **conserved adjacency** in two genomes \(G\) and \(H\) if either \(a\) and \(b\), or \(-b\) and \(-a\) are consecutive in both \(G\) and \(H\).

**Example:**

\[
\begin{align*}
G &= 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \\
H &= 0 \quad 3 \quad -2 \quad -1 \quad 4 \quad -5 \quad 6 \quad 7
\end{align*}
\]

**Property 1:** Upgrades easily to sets of \(k\) genomes.

**Property 2:** Invariant in optimal rearrangement scenarios.

**Property 3:** Independent of a model of evolution.

**Limits:** In larger sets of genomes, few adjacencies are completely conserved.
Adjacencies in mitochondrial genomes of *Arthropoda*

Fruit Fly

Silkworm

Tick
Adjacencies in mitochondrial genomes of *Arthropoda*
Conserved intervals

Definition:

A pair \([a, b]\) is a conserved interval in two genomes \(G\) and \(H\) if:
1) either \(a\) precedes \(b\), or \(-b\) precedes \(-a\), and
2) the sets of genes between \(a\) and \(b\) are the same.

Irreducible: Not the union of shorter conserved intervals.

Example:

\[
G = \begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 \\
H & = \begin{array}{cccccccc}
0 & \quad 3 & \quad -2 & \quad -1 & \quad 4 & \quad -5 & \quad 6 & \quad 7 \\
\end{array}
\]

Compact representation ("family portrait"):

\[
G' = \begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 \\
\]
\]
Conserved intervals

Definition:
A pair \([a, b]\) is a conserved interval in two genomes \(G\) and \(H\) if:
1) either \(a\) precedes \(b\), or \(-b\) precedes \(-a\), and
2) the sets of genes between \(a\) and \(b\) are the same.

Irreducible: Not the union of shorter conserved intervals.

Example:

\[
G' = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]

\[
H = \begin{array}{cccccccc}
0 & 3 & -2 & -1 & 4 & -5 & 6 & 7 \\
\end{array}
\]

Compact representation ("family portrait"):

\[
G' = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]
**Conserved intervals**

**Definition:**
A pair \([a, b]\) is a **conserved interval** in two genomes \(G\) and \(H\) if:
1) either \(a\) precedes \(b\), or \(−b\) precedes \(−a\), and
2) the sets of genes between \(a\) and \(b\) are the same.

**Irreducible:** Not the union of shorter conserved intervals.

**Example:**

\[
\begin{align*}
G &= 0 \ 1 \ 2 \ 3 \quad \boxed{4 \ 5 \ 6} \ 7 \\
H &= 0 \ 3 \ −2 \ −1 \quad \boxed{4 \ −5 \ 6} \ 7
\end{align*}
\]

**Compact representation** ("family portrait"):

\[
\begin{align*}
G &= \boxed{0 \ 1 \ 2 \ 3} \quad \boxed{4 \ 5 \ 6} \ 7
\end{align*}
\]
**Conserved intervals**

**Definition:**
A pair \([a, b]\) is a **conserved interval** in two genomes \(G\) and \(H\) if:
1) either \(a\) precedes \(b\), or \(-b\) precedes \(-a\), and
2) the sets of genes between \(a\) and \(b\) are the same.

**Irreducible:** Not the union of shorter conserved intervals.

**Example:**
\[
G = 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ \boxed{6 \ 7} \\
H = 0 \ 3 \ -2 \ -1 \ 4 \ -5 \ \boxed{6 \ 7}
\]

**Compact representation** ("family portrait"): 
\[
G' = \boxed{0 \ 1 \ 2 \ 3} \ \boxed{4 \ 5} \ \boxed{6 \ 7}
\]
Conserved intervals

Definition:
A pair \([a, b]\) is a conserved interval in two genomes \(G\) and \(H\) if:
1) either \(a\) precedes \(b\), or \(-b\) precedes \(-a\), and
2) the sets of genes between \(a\) and \(b\) are the same.

Irreducible: Not the union of shorter conserved intervals.

Example:

\[
G = 0 \quad \boxed{1 \ 2} \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \\
H = 0 \quad 3 \quad \boxed{-2 \ -1} \quad 4 \quad -5 \quad 6 \quad 7
\]

Compact representation ("family portrait"):

\[
G' = \boxed{0 \quad 1 \ 2} \quad 3 \quad 4 \quad \boxed{5 \quad 6 \ 7}
\]
Conserved intervals

**Definition:**
A pair \([a, b]\) is a **conserved interval** in two genomes \(G\) and \(H\) if:

1) either \(a\) precedes \(b\), or \(-b\) precedes \(-a\), and
2) the sets of genes between \(a\) and \(b\) are the same.

**Irreducible:** Not the union of shorter conserved intervals.

**Example:**

\[
\begin{align*}
G &= 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \\
H &= 0 \quad 3 \quad -2 \quad -1 \quad 4 \quad -5 \quad 6 \quad 7
\end{align*}
\]

**Compact representation** ("family portrait"): 

\[
\begin{align*}
G &= \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\end{align*}
\]
Conserved intervals in mitochondrial genomes of *Arthropoda*
Conserved intervals in mitochondrial genomes of *Arthropoda*
**Conserved intervals in mitochondrial genomes of *Arthropoda***

<table>
<thead>
<tr>
<th></th>
<th>Fruit Fly</th>
<th>Mosquito</th>
<th>Silkworm</th>
<th>Locust</th>
<th>Tick</th>
<th>Centipede</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conserved Intervals</strong></td>
<td>[Diagram]</td>
<td>[Diagram]</td>
<td>[Diagram]</td>
<td>[Diagram]</td>
<td>[Diagram]</td>
<td>[Diagram]</td>
</tr>
</tbody>
</table>
Conserved intervals in mitochondrial genomes of *Arthropoda*
Conserved intervals in mitochondrial genomes of *Arthropoda*
Properties

Property 1: Upgrades easily to sets of $k$ genomes.

Property 2: Invariant in (most) optimal rearrangement scenarios.

Property 3: Independent of a model of evolution.

Property 4: Computable in linear time:

1: stack 0 on $S$, stack $n$ on $M$
2: $M_0 \leftarrow n$
3: for $i = 1, \ldots, n$ do
4: unstack from $M$ all elements $m$ smaller than $|\pi_i|$
5: $M_i \leftarrow m$
6: stack the element $|\pi_i|$ on $M$
7: unstack from $S$ all indices $s$ such that ($|\pi_i| < \pi_s$ or $|\pi_i| > M_s$)
8: if $i - s = \pi_i - \pi_s$ and $M_i = M_s$ then
9: output positive irreducible conserved interval $[\pi_s, \pi_i]$
10: end if
11: if $\pi_i$ is positive then
12: stack the index $i$ on $S$
13: end if
14: end for
Algorithm summary

Two permutations:

- find all irreducible conserved intervals in $\mathcal{O}(n)$ time and space
- find all $K$ conserved intervals in $\mathcal{O}(n + K)$ time and $\mathcal{O}(n)$ space

More than two permutations:

- find the intersection of two sets of irreducible intervals in $\mathcal{O}(n)$ time and space
- find all irreducible conserved of a set of $k$ permutations in $\mathcal{O}(kn)$ time and $\mathcal{O}(n)$ space
Similarity and distance

The number of conserved intervals between two genomes is a measure of similarity.

It is possible to derive a measure of distance between two genomes:

\[ d(G, H) = N_1 + N_2 - 2N \]

where

- \( N_1 \) is the number of conserved intervals in \( G \)
- \( N_2 \) is the number of conserved intervals in \( H \)
- \( N \) is the number of conserved intervals in \( G \cup H \)
Interval distance and reversal/transposition distance table

<table>
<thead>
<tr>
<th></th>
<th>Fruit Fly</th>
<th>Mosquito</th>
<th>Silkworm</th>
<th>Locust</th>
<th>Tick</th>
<th>Centipede</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fruit Fly</td>
<td>–</td>
<td>90</td>
<td>2</td>
<td>62</td>
<td>1</td>
<td>158</td>
</tr>
<tr>
<td>Mosquito</td>
<td>90</td>
<td>2</td>
<td>–</td>
<td>–</td>
<td>140</td>
<td>3</td>
</tr>
<tr>
<td>Silkworm</td>
<td>62</td>
<td>1</td>
<td>140</td>
<td>3</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Locust</td>
<td>62</td>
<td>1</td>
<td>140</td>
<td>3</td>
<td>116</td>
<td>2</td>
</tr>
<tr>
<td>Tick</td>
<td>158</td>
<td>2</td>
<td>200</td>
<td>4</td>
<td>180</td>
<td>3</td>
</tr>
<tr>
<td>Centipede</td>
<td>188</td>
<td>3</td>
<td>230</td>
<td>5</td>
<td>194</td>
<td>4</td>
</tr>
</tbody>
</table>
Links with rearrangement theories

**Link 1:** Conserved intervals between two permutations are the *connected components* of the *interleaving cycles* of the *breakpoint graph*. (First noticed by Hannenhalli, 1995.)

**Link 2:** Interval distance is sensitive to the length of rearranged segments.

**Link 3:** Optimal rearrangement scenarios that break conserved intervals are suspicious.
Overview: New models and algorithms for genome comparison

- Introduction
  - Comparative genomics
  - Gene clusters

- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results

- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes

- Summary and Conclusion
Summary: Gene clusters and common intervals

Some algorithmic results:

• Find all common intervals of $k$ permutations in $\mathcal{O}(kn + |\text{output}|)$ time.

• Find all common intervals of $k$ sequences in $\mathcal{O}(kn^2)$ time.

• Find all conserved intervals of $k$ permutations in $\mathcal{O}(kn)$ time
Conclusion

Points raised:

• Comparative genomics can help in functional genome annotation
• Conserved regions in genomes have a static and a dynamic aspect
• Interesting combinatorics in Bioinformatics

Next steps:

• Statistical assessment of gene clusters
• Patterns in overlapping gene clusters
• Application to more data
Acknowledgments

Common intervals

- Steffen Heber (Raleigh)
- Mathieu Raffinot (Paris)
- Hannes Luz (Berlin)
- Thomas Schmidt (Bielefeld)

Conserved intervals

- Anne Bergeron (Montréal)

References:
