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 ... 10,000 kilobases (KB)
  - genes: $\approx$ 450 ... 10,000 open reading frames (ORFs)
- 18 archaeal genomes (first: *M. janaschii*, 1996)
  - size: $\approx$ 1,500 ... 6,000 KB
  - genes: $\approx$ 1,500 ... 4,500 ORFs
- 23 eukaryal genomes (first: *S. cerevisiae*, 1997)
  - size: $\approx$ 12,000 ... 2,800,000 KB
  - genes: $\approx$ 6,000 ... 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 ... 10,000 kilobases (KB)
  - genes: $\approx$ 450 ... 10,000 open reading frames (ORFs)
- 18 archaeal genomes (first: *M. janaschii*, 1996)
  - size: $\approx$ 1,500 ... 6,000 KB
  - genes: $\approx$ 1,500 ... 4,500 ORFs
- 23 eukaryal genomes (first: *S. cerevisiae*, 1997)
  - size: $\approx$ 12,000 ... 2,800,000 KB
  - genes: $\approx$ 6,000 ... 50,000 ORFs

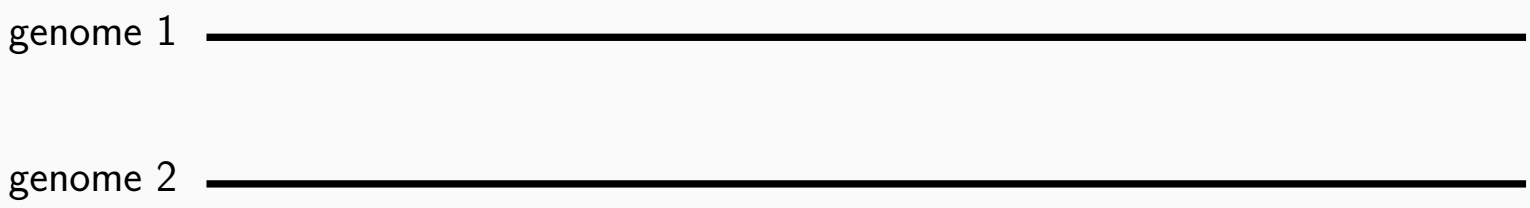
Next steps:

functional genomics (transcriptomics, proteomics, metabolomics, ...)
comparative genomics

Comparative genomics "at a higher level"

Concentrate on large scale layout of the genomes:

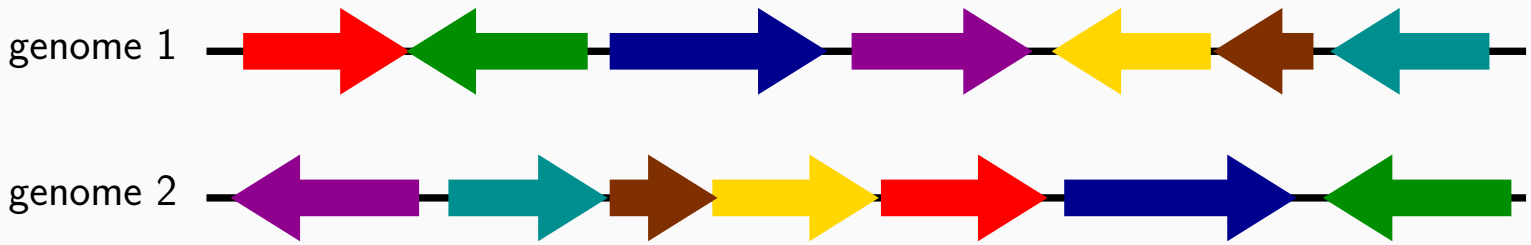
- Study genomes based on their *gene order*.
- Represent genomes by their sequence of genes.



Comparative genomics "at a higher level"

Concentrate on large scale layout of the genomes:

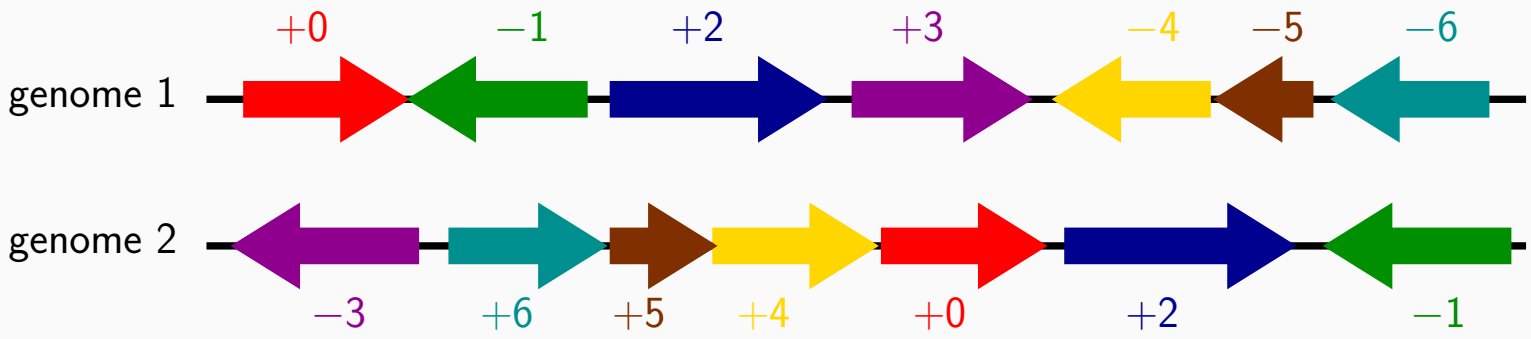
- Study genomes based on their *gene order*.
- Represent genomes by their sequence of genes.



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.



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 $+0\ \boxed{+1\ +2\ +3\ +4}\ +5 \longrightarrow +0\ \boxed{-4\ -3\ -2\ -1}\ +5$

transposition $0\ \boxed{1\ \ 2}\ 3\ \ 4\ \ 5 \longrightarrow 0\ \ 3\ \ 4\ \ \boxed{1\ \ 2}\ \ 5$

Resulting distances and problems:

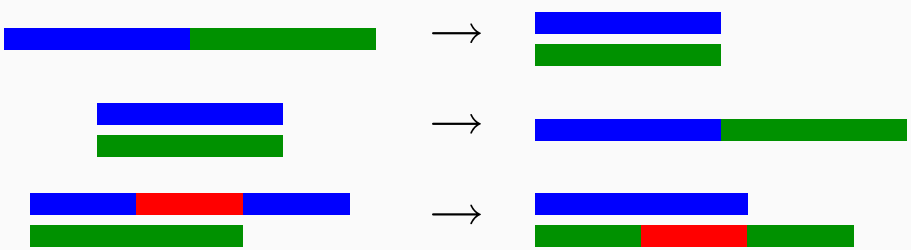
- (signed) reversal distance → sorting (signed) permutations by reversals
- transposition distance → sorting permutations by transpositions

Generalization: multiple chromosomes

→ additional operations: fission

fusion

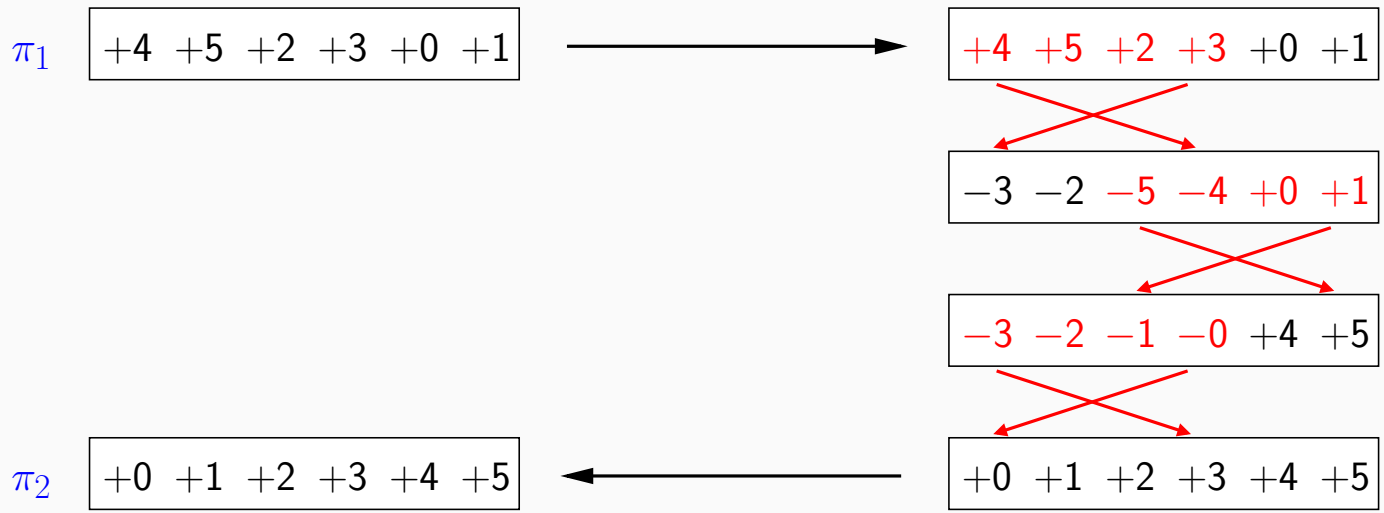
translocation



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

Sorting by reversals

Problem: Given two (signed) permutations (genomes) π_1 and π_2 of the elements (genes) of the set $N = \{0, 1, \ldots, n\}$, find the minimal number of *reversals* that are necessary to transform π_1 into π_2.



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
($\approx 1 - 2$ years, $\approx 1 - 2$ 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
($\approx 1 - 2$ years, $\approx 1 - 2$ 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
($\approx 1 - 2$ 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)
- 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:

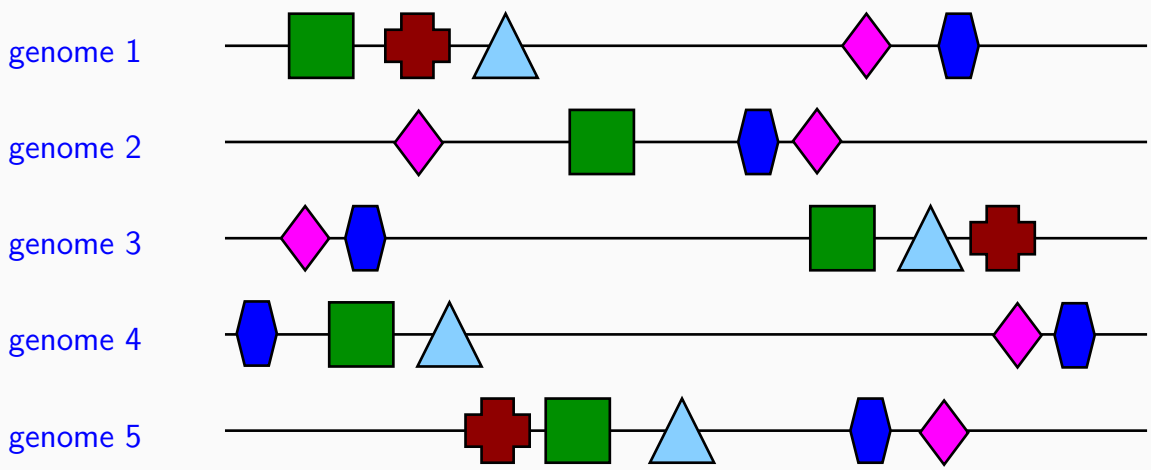
- interacting proteins
- proteins of the same protein complex
- enzymes of the same metabolic pathway

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

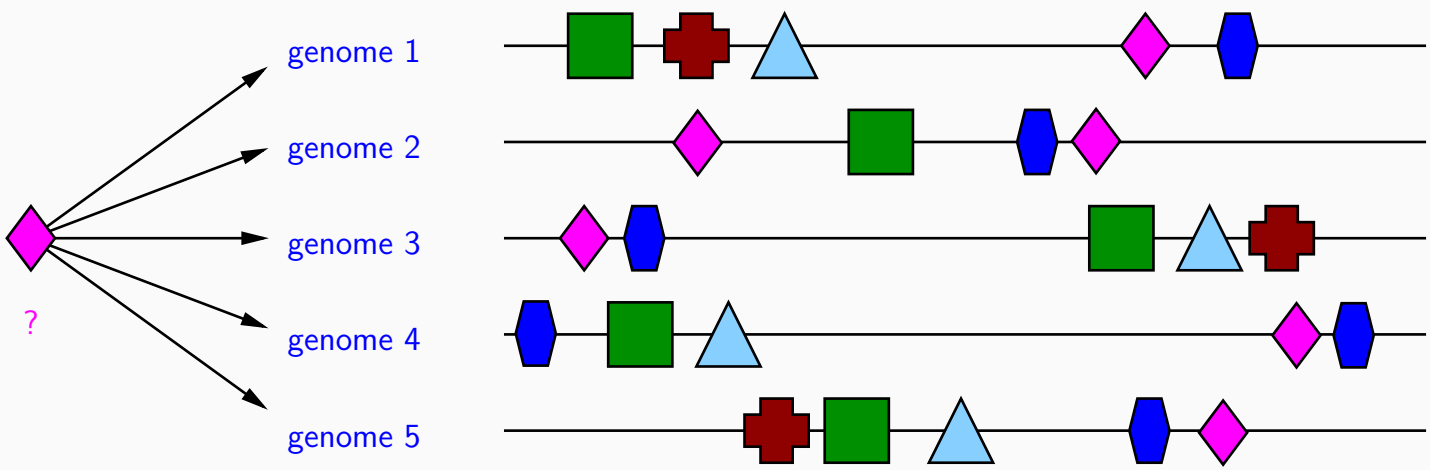


Genome-based gene function prediction

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

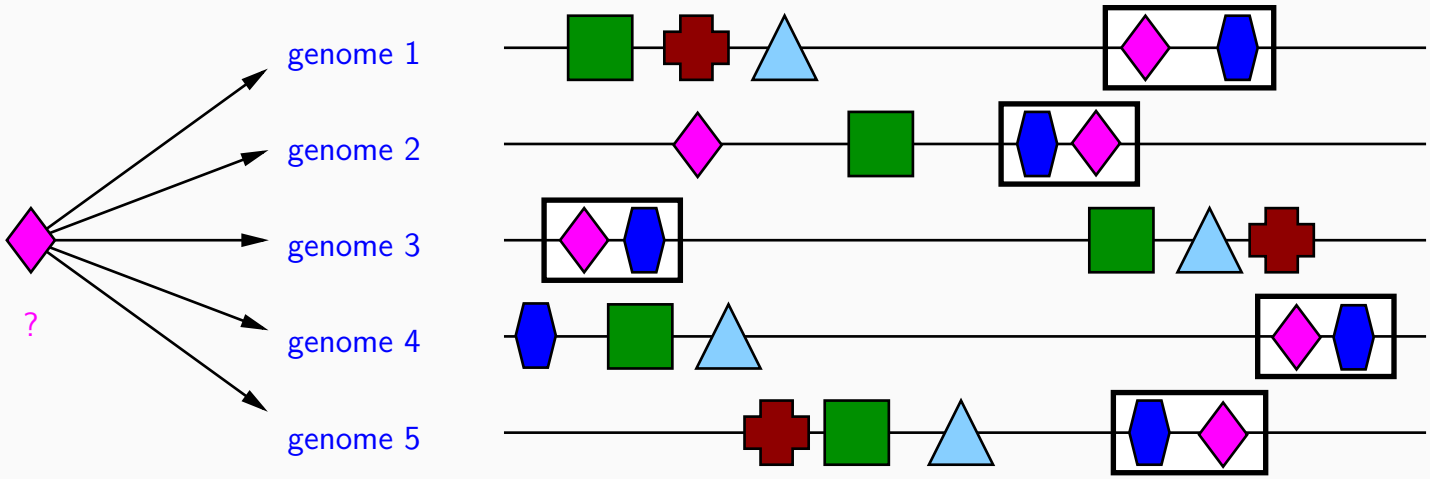


Genome-based gene function prediction

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

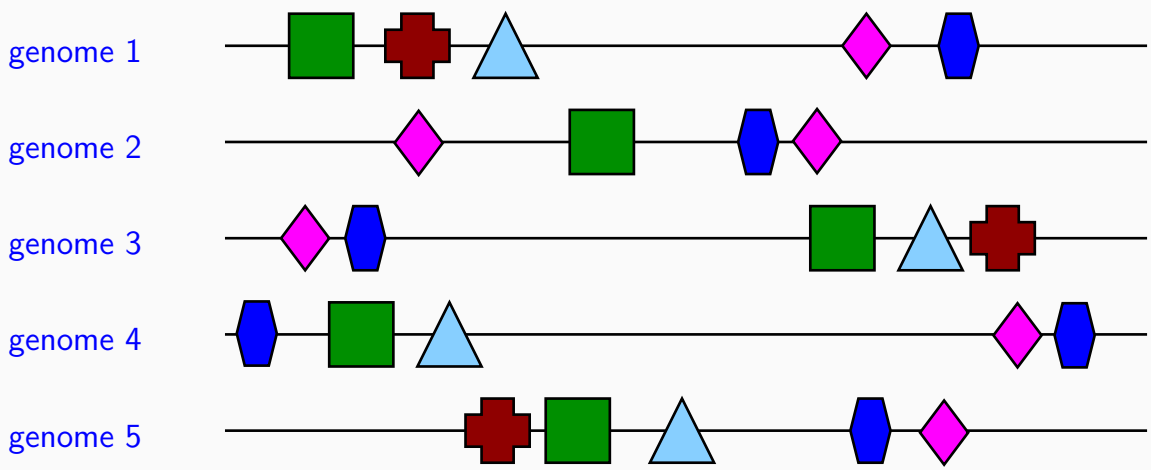


Genome-based gene function prediction

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

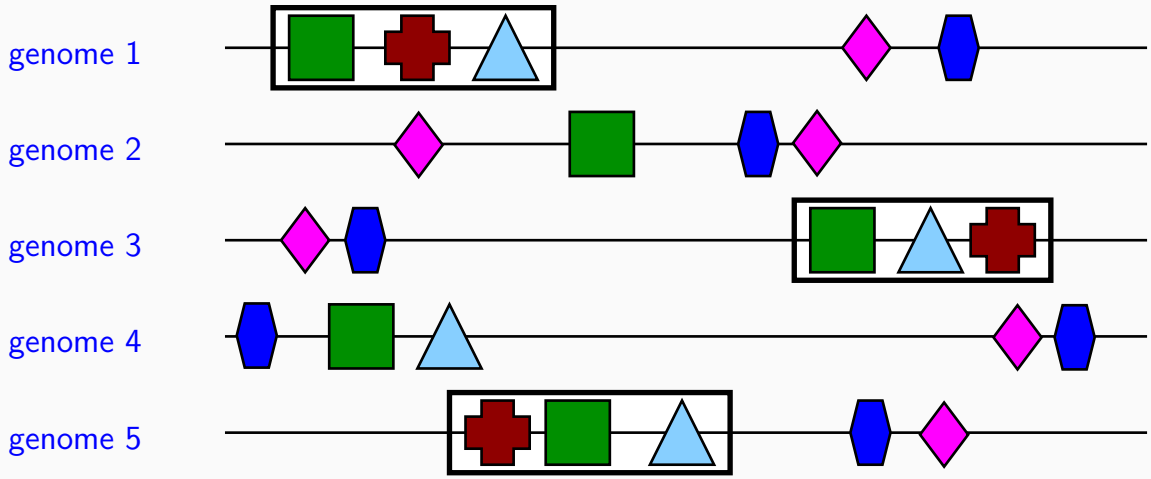


Genome-based gene function prediction

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

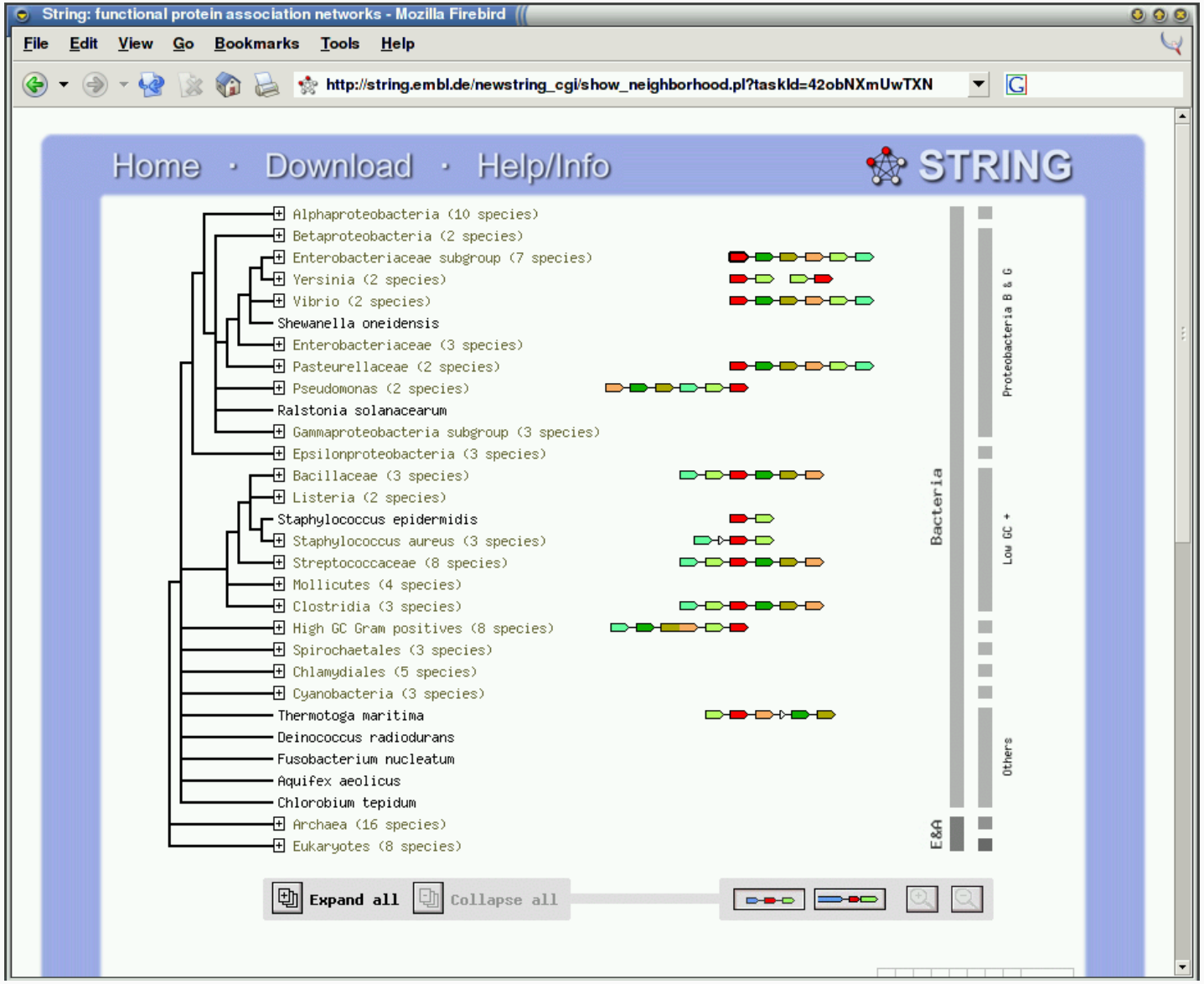
Marcotte *et al.*: Detecting protein function and protein-protein interactions from genome sequences. *Science*, 1999.

Overbeek *et al.*: The use of gene clusters to infer functional coupling. *PNAS*, 1999.

Snel *et al.*: STRING: A web-server to retrieve and display the repeatedly occurring neighbourhood of a gene, *NAR*, 2000.

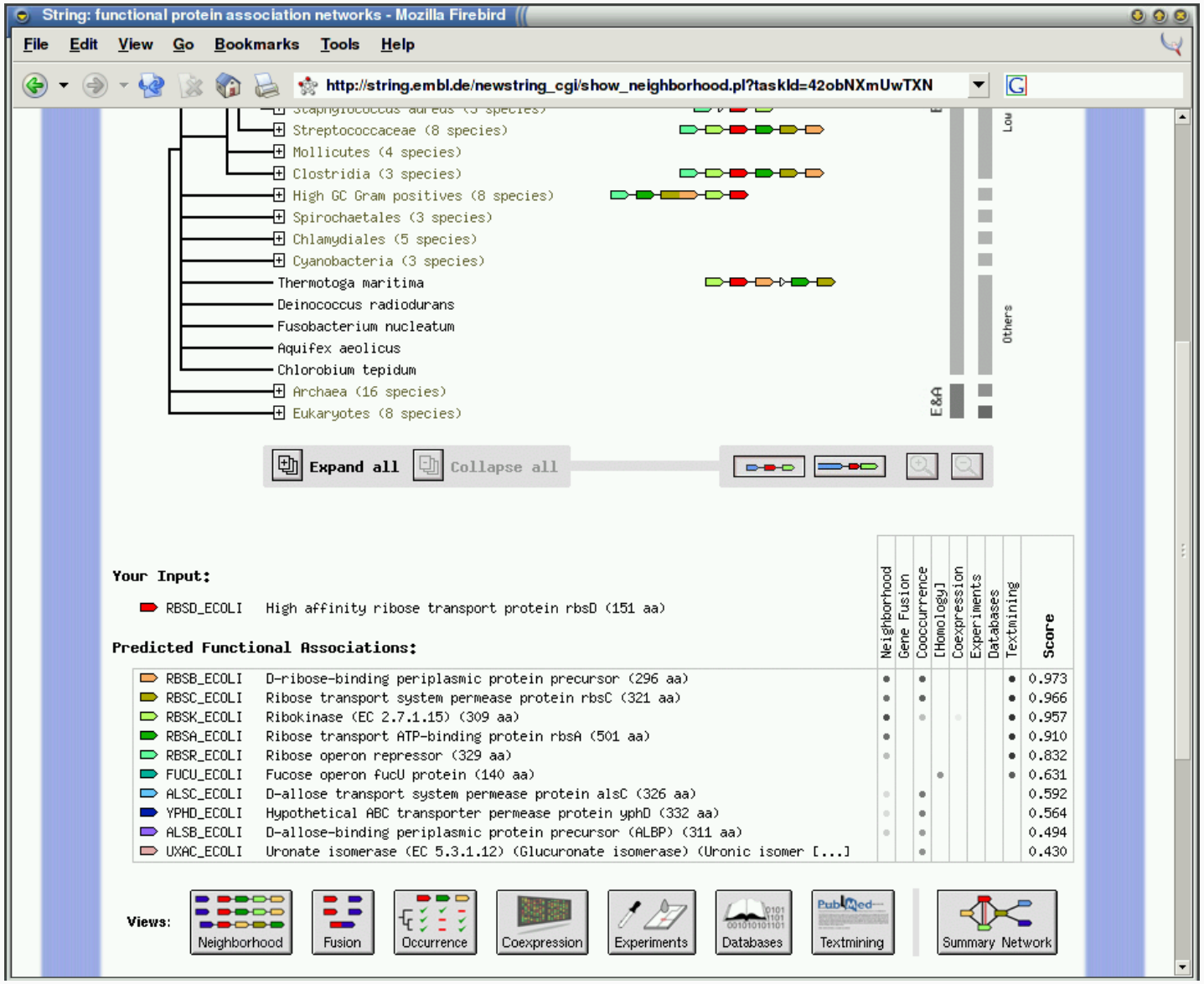
STRING Web server (Snel *et al.*, 2000)

http://string.embl.de/

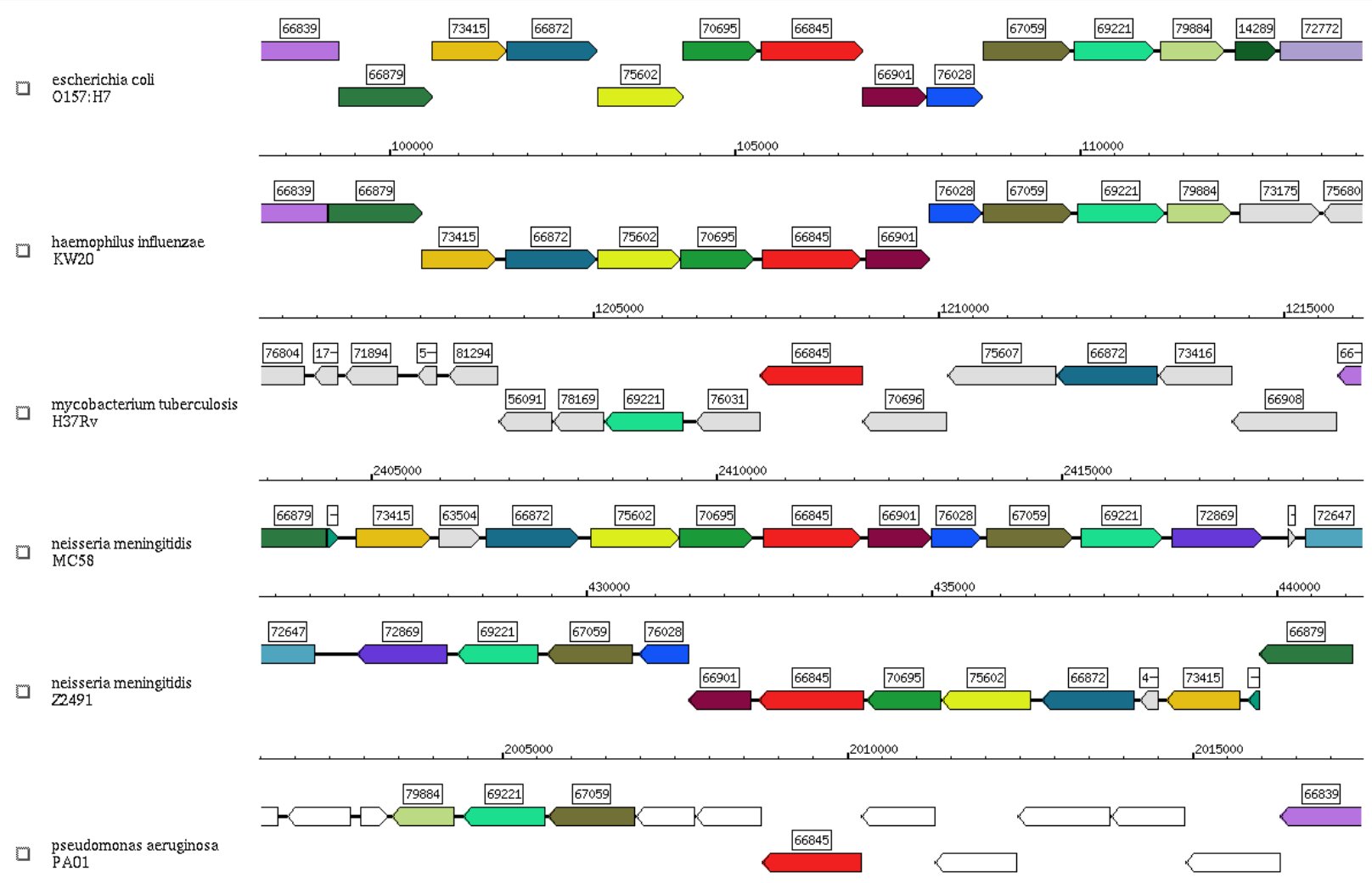


STRING Web server (Snel *et al.*, 2000)

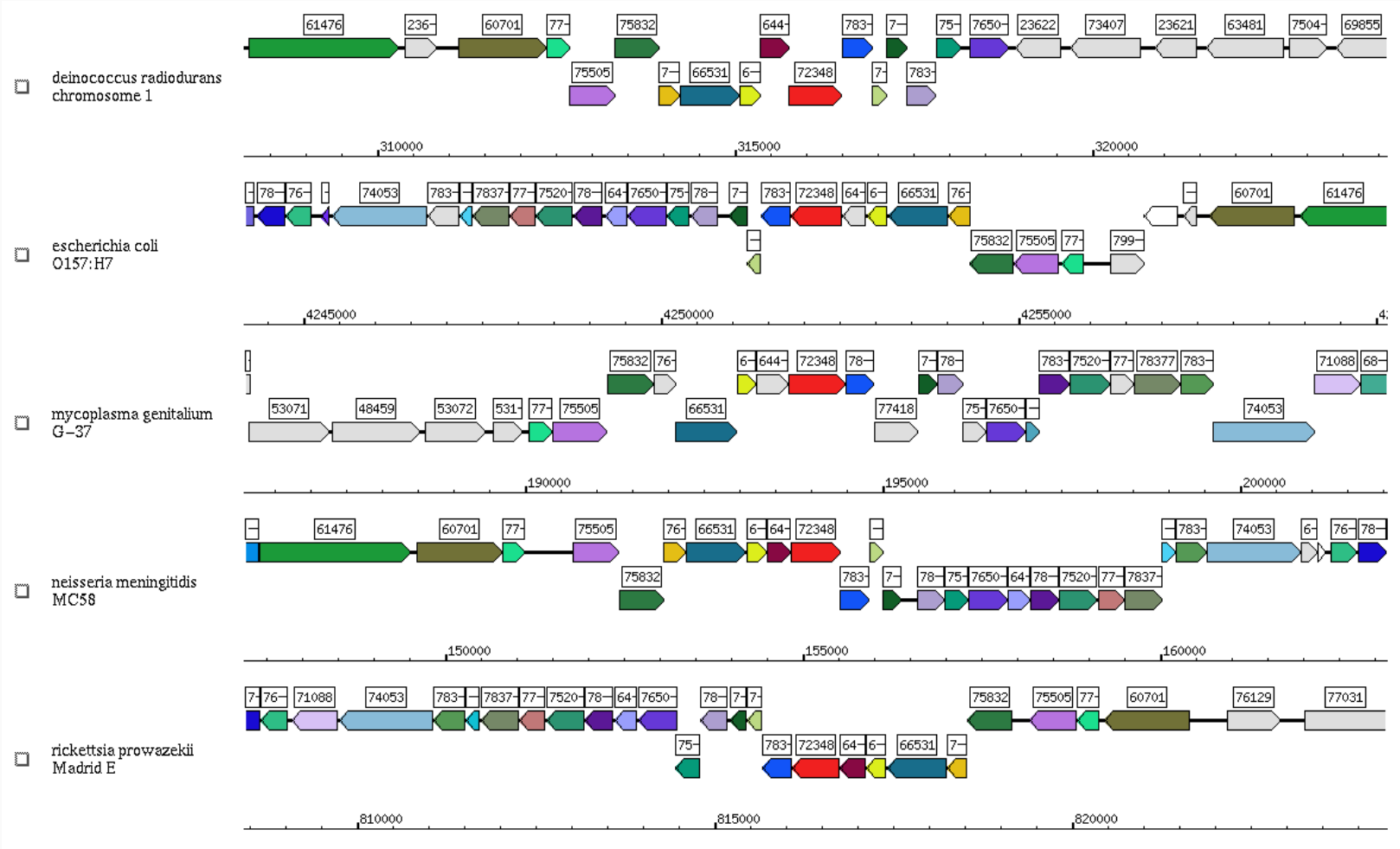
http://string.embl.de/



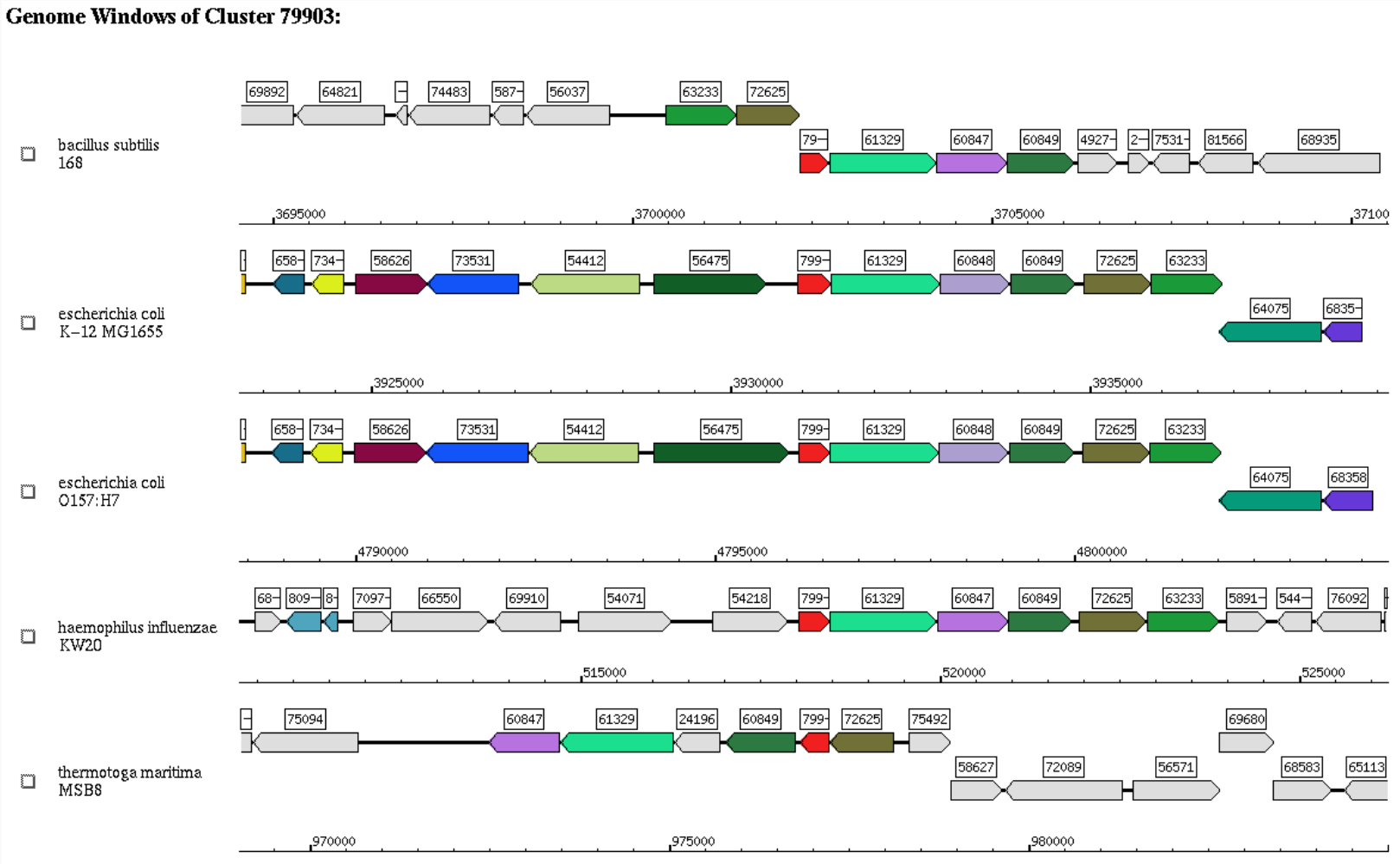
Genome Windows: DCW cluster (division and cell wall)



Genome Windows: Ribosomal Super Operon



Genome Windows: Ribose-ABC-Transporter



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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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

Finding all commmon intervals of 2 permutations π_1 and π_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:

π_1	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
	0	1	2	3	4	5	6	7

Finding all commmon intervals of 2 permutations π_1 and π_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:

π_1	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
	0	1	2	3	4	5	6	7

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

Finding all commmon intervals of 2 permutations π_1 and π_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:

π_1	0	1	2	3	4	5	6	7	$f(1,4) = 4 - 1 - (6-3) = 0$
	0	1	2	3	4	5	6	7	
π_2	6	7	5	1	4	3	2	0	$f(1,4) = 7 - 1 - (4-1) = 3 > 0$
	0	1	2	3	4	5	6	7	

Finding all commmon intervals of 2 permutations π_1 and π_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:

π_1	0	1	2	3	4	5	6	7	$f(1,4) = 4 - 1 - (6-3) = 0$
	0	1	2	3	4	5	6	7	
π_2	6	7	5	1	4	3	2	0	$f(1,4) = 7 - 1 - (4-1) = 3 > 0$
	0	1	2	3	4	5	6	7	

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 π_1 and π_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 $x' \leq x$:

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

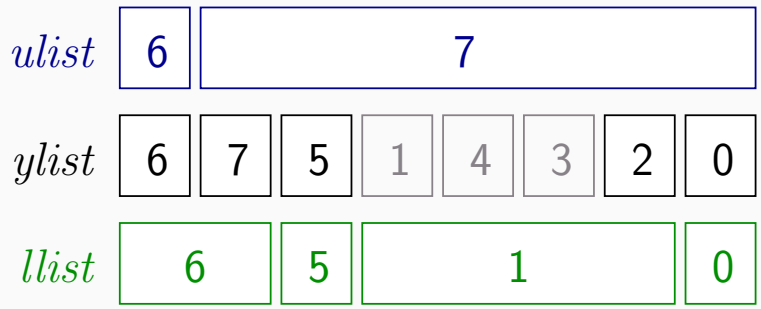
Lemma:
For fixed x, $f(x, y)$ increases monotonically for the non-wasteful indices y $(> x)$.

Algorithm (Idea):

- x runs in right-to-left direction through a doubly linked list $ylist$ that initially contains the entries of π_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 $f(x, y) = 0$.

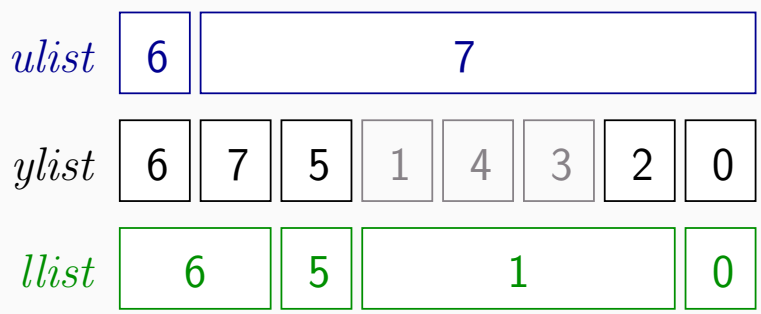
Algorithm RC (Uno & Yagiura)

- 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.



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:
$\mathcal{O}(n + |\text{output}|)$ 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 π_1 and π_j.
Output all intervals that are found in all of these comparisons.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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 π_1 and π_j.
Output all intervals that are found in all of these comparisons.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

[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 π_1 and π_j.
Output all intervals that are found in all of these comparisons.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

[0,4] [0,5] [0,7] [1,4] [1,5] [1,7] [2,3] [2,4]
[3,4] [5,7] [6,7]

[0,1] [0,2] [0,4] [0,5] [0,6] [0,7] [1,2] [1,4]
[1,5] [1,6] [2,4] [2,5] [2,6] [3,4] [3,5] [3,6]
[5,6]

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 π_1 and π_j.
Output all intervals that are found in all of these comparisons.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

[0,4] [0,5] [0,7] [1,4] [1,5] [1,7] [2,3] [2,4]
[3,4] [5,7] [6,7]

[0,1] [0,2] [0,4] [0,5] [0,6] [0,7] [1,2] [1,4]
[1,5] [1,6] [2,4] [2,5] [2,6] [3,4] [3,5] [3,6]
[5,6]

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 π_1 and π_j.
Output all intervals that are found in all of these comparisons.

π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

[0,4] [0,5] [0,7] [1,4] [1,5] [1,7] [2,3] [2,4] [3,4] [5,7] [6,7]

[0,1] [0,2] [0,4] [0,5] [0,6] [0,7] [1,2] [1,4] [1,5] [1,6] [2,4] [2,5] [2,6] [3,4] [3,5] [3,6] [5,6]

Analysis:

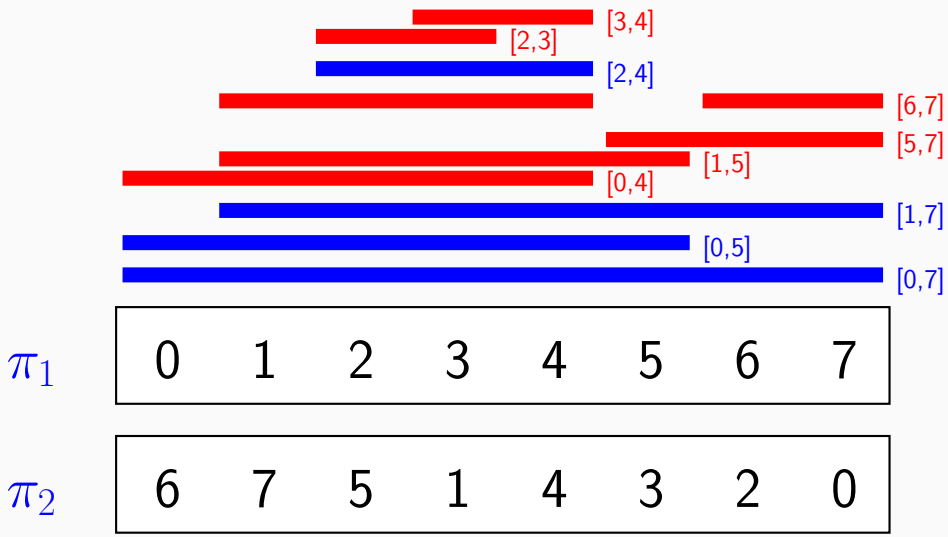
$\mathcal{O}(kn + \sum |K_i|)$ time

where $K_i =$ the number of common intervals of π_1 and π_i.

Irreducible Intervals

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

Observation: Common intervals form "chains" of non-trivially overlapping intervals.



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

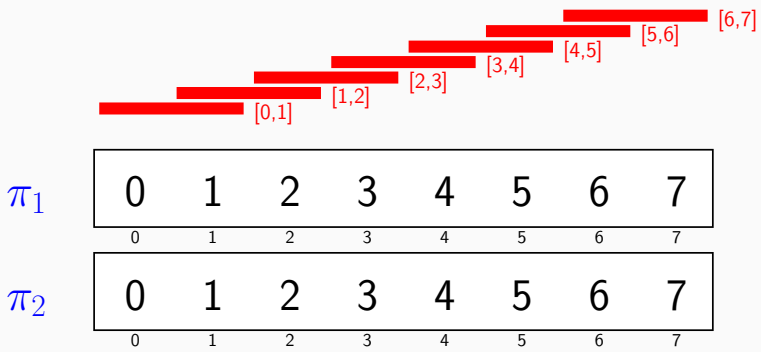
Lemma:
The subchains of all the maximal chains of irreducible intervals generate exactly all common intervals.

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

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

Example:

$K = 1$

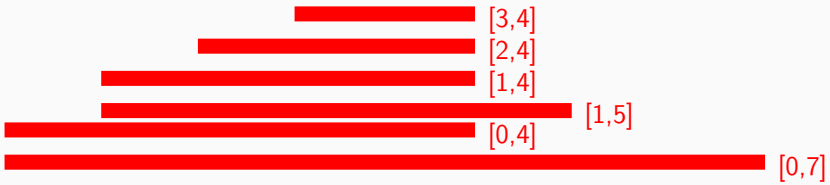


$K = n - 1$

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.

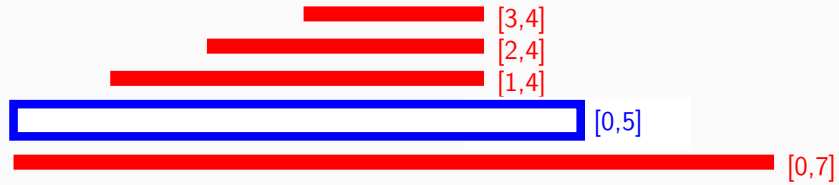


π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.

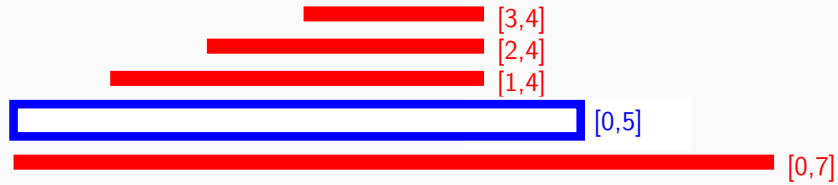


π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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.



π_1	0	1	2	3	4	5	6	7
π_2	6	7	5	1	4	3	2	0
π_3	7	0	1	2	4	3	5	6

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
   $\Rightarrow$ multichromosomal permutations

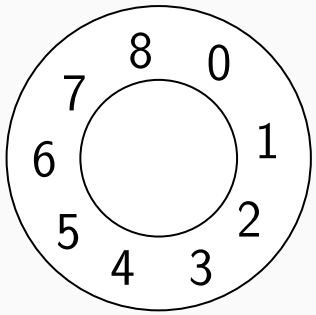
π_1	0	1	2	3	4	5	6	7	8
π_2	5	1	0	2	3	4	6	8	7

2. Genes of a cluster should lie on the same DNA strand
   $\Rightarrow$ signed permutations

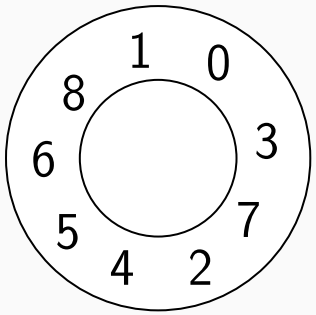
π_1	$+0$	$+1$	$+2$	$+3$	$+4$	$+5$	$+6$	$+7$	$+8$
π_2	$+8$	$+7$	$+3$	$+5$	$+4$	-6	-0	-1	-2

3. Bacterial, archaeal, and mitochondrial DNA is often circular
   $\Rightarrow$ circular permutations

π_1



π_1



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.
$\rightarrow$ genomes are modeled as sequences instead of permutations.

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Common intervals: $\{3\}$

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Common intervals: $\{3\}$ $\{1\}$

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Common intervals: $\{3\}$ $\{1\}$ $\{5\}$

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Common intervals: {3} {1} {5} {1,5}

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Common intervals: {3} {1} {5} {1,5} {1,3,5}

Formal model

Given: k sequences $\mathcal{S} = (S_1, S_2, \ldots, S_k)$ over an alphabet Σ.

Common interval:
a subset $C \subseteq \Sigma$ whose elements occur contiguously in each $S_l \in \mathcal{S}$.

Goal:
Find all maximal occurrences of common intervals in $\mathcal{S}$.

Example:

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		

Common intervals: {3} {1} {5} {1,5} {1,3,5}

An elementary algorithm for two sequences

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

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

$NUM(i, j)$:

$i \backslash j$	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
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

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

S_1	3	1	2	3	1	5	2	6
	0	1	2	3	4	5	6	7

S_2	4	3	5	5	5	1	4	2	2

An elementary algorithm for two sequences

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

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

$NUM(i, j)$:

$i \backslash j$	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
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

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

S_1

3	1	2	3	1	5	2	6
0	1	2	3	4	5	6	7

S_2

4	3	5	5	5	1	4	2	2

$i = j$

An elementary algorithm for two sequences

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

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

$NUM(i, j)$:

$i \backslash j$	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
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

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

S_1	3	1	2	3	1	5	2	6
	0	1	2	3	4	5	6	7

S_2	4	3	5	5	5	1	4	2	2
		i	j						

An elementary algorithm for two sequences

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

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

$NUM(i, j)$:

$i \backslash j$	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
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

Algorithm:

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

S_1

3	1	2	3	1	5	2	6
0	1	2	3	4	5	6	7

S_2

4	3	5	5	5	1	4	2	2
	i		j					

An elementary algorithm for two sequences

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

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

$NUM(i, j)$:

$i \backslash j$	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
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

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

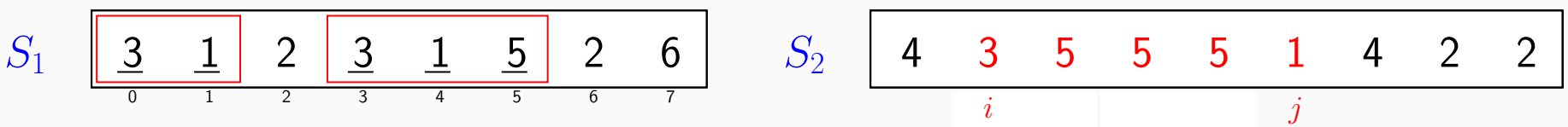
Preprocessing: compute two tables for $S_1 = (3, 1, 2, 3, 1, 5, 2, 6)$:

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

$NUM(i, j)$:

$i \backslash j$	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
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

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

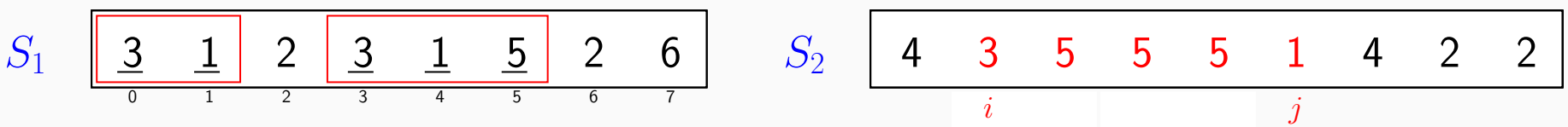
Preprocessing: compute two tables for $S_1 = (3, 1, 2, 3, 1, 5, 2, 6)$:

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

$NUM(i, j)$:

$i \backslash j$	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
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

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



Analysis: $\mathcal{O}(n^2)$ 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 $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space.

More than two sequences:

- Find all common intervals in k sequences in $\mathcal{O}(kn^2)$ time and space.
- Find all common intervals that appear in at least k' out of k given sequences in $\mathcal{O}(k(1+k-k')n^2)$ time and $\mathcal{O}(kn^2)$ space.

Overview: New models and algorithms for genome comparison

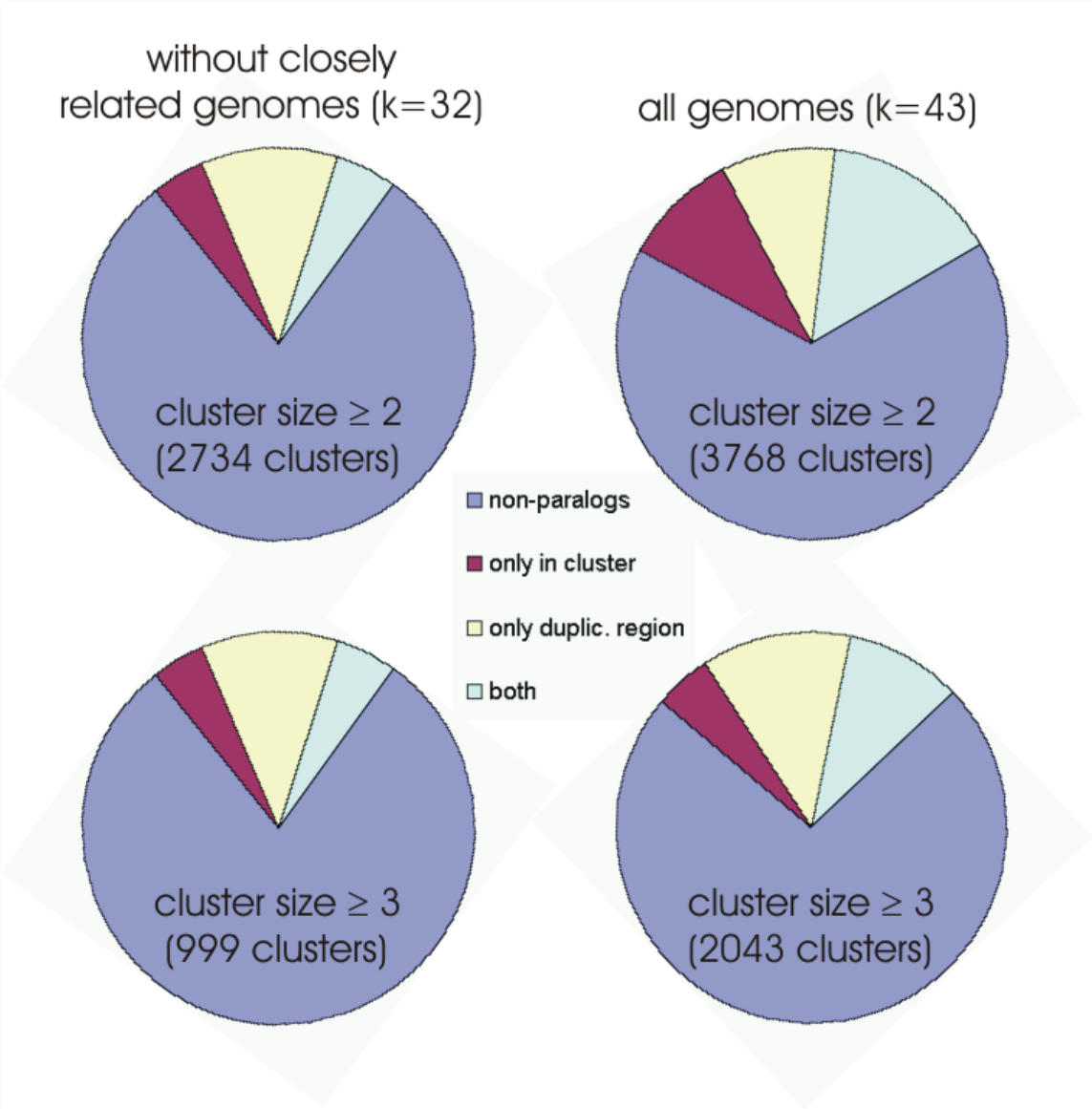
- Introduction
  - Comparative genomics
  - Gene clusters
- Finding gene clusters
  - Gene clusters of permutations
  - Gene clusters of sequences
  - Experimental results
- Conserved intervals
  - Finding conserved intervals
  - Application to mitochondrial genomes
- Summary and Conclusion

Experimental results. Data source: COG

Aquifex aeolicus complete genome - 0..1551335
1529 proteins

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

Experimental results. Application to 43 bacterial genomes



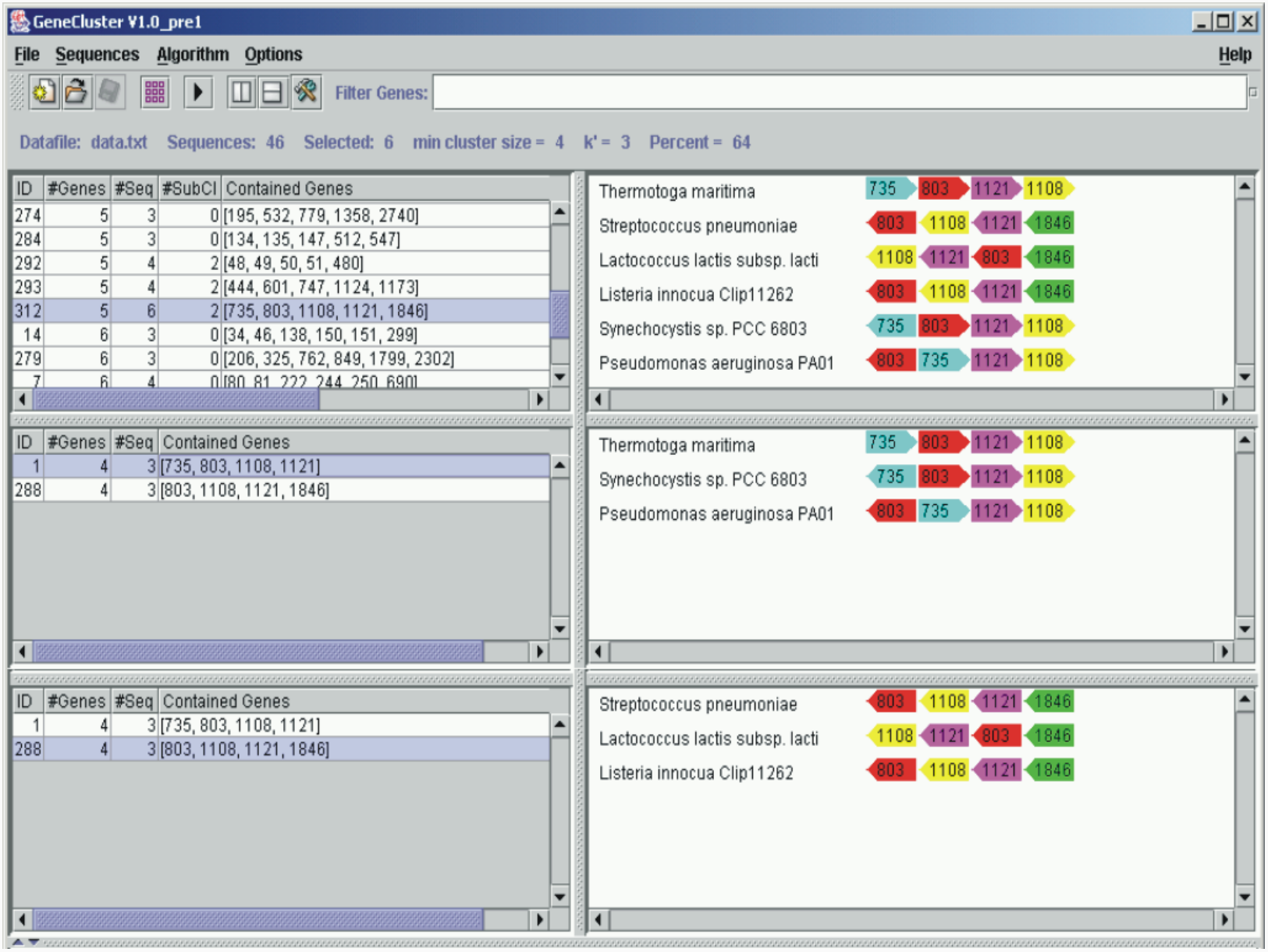
Experimental results. Graphical inspection of gene clusters

Text output of algorithm

```
S1 : Mycoplasma genitalium
S2 : Mycoplasma pneumoniae
S3 : Mycoplasma pulmonis


Basic Algorithm Output
7: #9# *2* S1: (1,9) S2: (1,9)  Genes: [84, 125, 172, 187, 188, 470, 486, 592, 2214]
15: #2# *3* S1: (3,4) (205,206) S2: (3,4) (122,123) S3: (372,373)  Genes: [187, 188]
30: #2# *3* S1: (6,7) S2: (6,7) S3: (52,53)  Genes: [125, 470]
35: #2# *3* S1: (8,9) S2: (8,9) S3: (13,14)  Genes: [84, 486]
38: #4# *2* S1: (10,15) S2: (14,19)  Genes: [189, 190, 358, 1132]
47: #7# *2* S1: (18,24) S2: (20,26)  Genes: [12, 143, 191, 484, 553, 596, 3343]
64: #3# *2* S1: (25,27) S2: (28,30)  Genes: [231, 463, 781]
67: #3# *2* S1: (29,31) S2: (32,34)  Genes: [35, 693, 2176]
72: #5# *2* S1: (33,37) S2: (43,47)  Genes: [124, 173, 580, 1435, 1488]
76: #2# *3* S1: (35,36) S2: (45,46) S3: (332,333)  Genes: [124, 173]
81: #4# *2* S1: (38,41) S2: (50,53)  Genes: [554, 579, 1744, 1925]
87: #4# *3* S1: (42,45) S2: (55,58) S3: (422,425)  Genes: [687, 1176, 1177, 3842]
98: #15# *2* S1: (42,56) S2: (55,69)  Genes: [192, 213, 250, 267, 274, 295, 533, 541, 687, 690, 813, 1109, 117
188: #3# *3* S1: (54,56) S2: (67,69) S3: (180,182)  Genes: [250, 267, 690]
194: #6# *2* S1: (58,63) S2: (71,77)  Genes: [313, 462, 463, 477, 691, 1658]
205: #2# *2* S1: (67,68) S2: (81,82)  Genes: [21, 1136]
209: #5# *2* S1: (71,75) S2: (207,211)  Genes: [52, 474, 556, 653, 2190]
217: #3# *3* S1: (79,81) S2: (215,217) S3: (283,285) (410,412)  Genes: [444, 601, 1173]
218: #4# *3* S1: (79,82) S2: (215,218) S3: (283,286)  Genes: [444, 601, 1173, 4608]
232: #18# *2* S1: (79,96) S2: (215,232)  Genes: [37, 48, 49, 80, 81, 193, 238, 305, 359, 360, 444, 480, 601, 62
278: #2# *3* S1: (83,84) S2: (219,220) S3: (9,10)  Genes: [80, 81]
303: #2# *3* S1: (85,86) S2: (221,222) S3: (780,781)  Genes: [37, 193]
324: #2# *3* S1: (87,88) S2: (223,224) S3: (711,712)  Genes: [682, 1493]
342: #3# *3* S1: (89,91) S2: (225,227) S3: (428,430)  Genes: [48, 49, 480]
360: #3# *3* S1: (92,94) S2: (228,230) S3: (609,611)  Genes: [238, 360, 629]
368: #2# *3* S1: (95,96) S2: (231,232) S3: (515,516)  Genes: [305, 359]
371: #4# *2* S1: (99,102) S2: (235,238)  Genes: [64, 154, 692, 721]
374: #2# *3* S1: (101,102) S2: (237,238) S3: (200,201)  Genes: [64, 154]
```

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 geome 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.

History (partial): Sankoff 1992; Hannenhalli & Pevzner 1995; Bafna & Pevzner 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; Heber & Stoye 2001; Bergeron, Heber & Stoye 2002.

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:

$$\begin{array}{ccccccccc} G & = & 0 & \boxed{1 \quad 2} & 3 & 4 & 5 & \boxed{6 \quad 7} \\ H & = & 0 & 3 & \boxed{-2 \quad -1} & 4 & -5 & \boxed{6 \quad 7} \end{array}$$

Property 1: Upgrades easily to sets of k genomes.

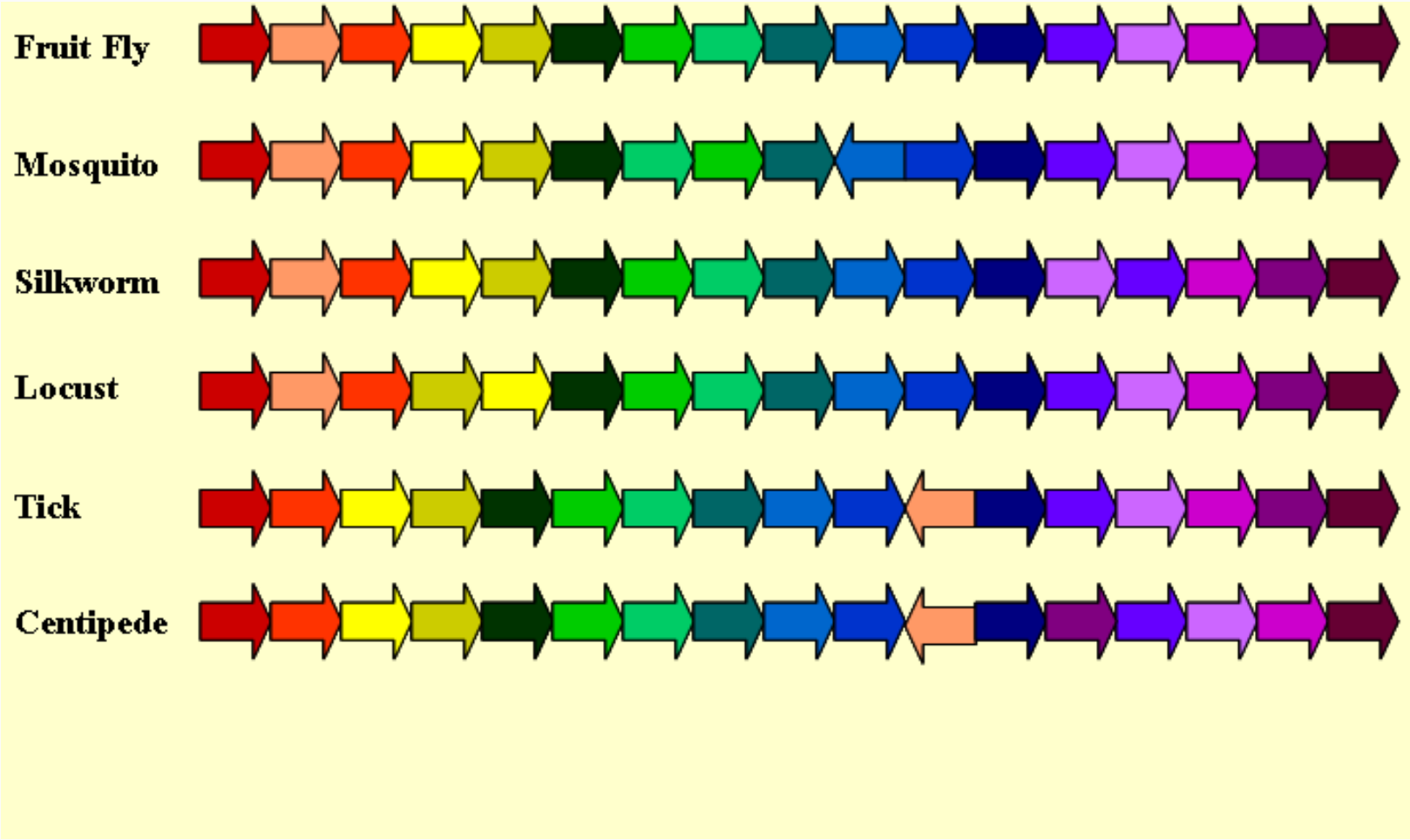
Property 2: Invariant in optimal rearrangement scenarios.

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*

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:

$$\begin{array}{ccccccccc} G & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ H & = & 0 & 3 & -2 & -1 & 4 & -5 & 6 & 7 \end{array}$$

Compact representation ("family portrait"):

$$\begin{array}{ccccccccc} G & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

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:

$$G = \boxed{0 \quad 1 \quad 2 \quad 3 \quad 4} \quad 5 \quad 6 \quad 7$$

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

Compact representation ("family portrait"):

$$G = \boxed{0} \; \boxed{1 \quad 2 \quad 3} \; \boxed{4} \quad 5 \quad 6 \quad 7$$

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:

$$G = 0 \quad 1 \quad 2 \quad 3 \quad \boxed{4 \quad 5 \quad 6} \quad 7$$

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

Compact representation ("family portrait"):

$$G = \boxed{0} \quad \boxed{1 \quad 2 \quad 3} \quad \boxed{4} \quad \boxed{5} \quad \boxed{6} \quad 7$$

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:

$$G = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad \boxed{6 \quad 7}$$

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

Compact representation ("family portrait"):

$$G = \boxed{0} \; \boxed{1 \quad 2 \quad 3} \; \boxed{4} \; \boxed{5} \; \boxed{6} \; \boxed{7}$$

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:

$$G = 0 \quad \boxed{1 \quad 2} \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$$

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

Compact representation ("family portrait"):

$$G = \boxed{0} \quad \boxed{\boxed{1 \quad 2} \quad 3} \quad \boxed{4} \quad \boxed{5} \quad \boxed{6} \quad \boxed{7}$$

Conserved intervals

Definition:

A pair $[a, b]$ is a conserved interval in two genomes G and H if:

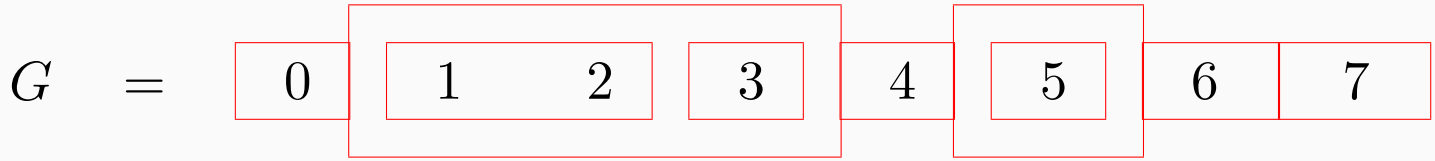
1) either a precedes b, or $-b$ precedes $-a$, and
2) the sets of genes between a and b are the same.

Irreducible: Not the union of shorter conserved intervals.

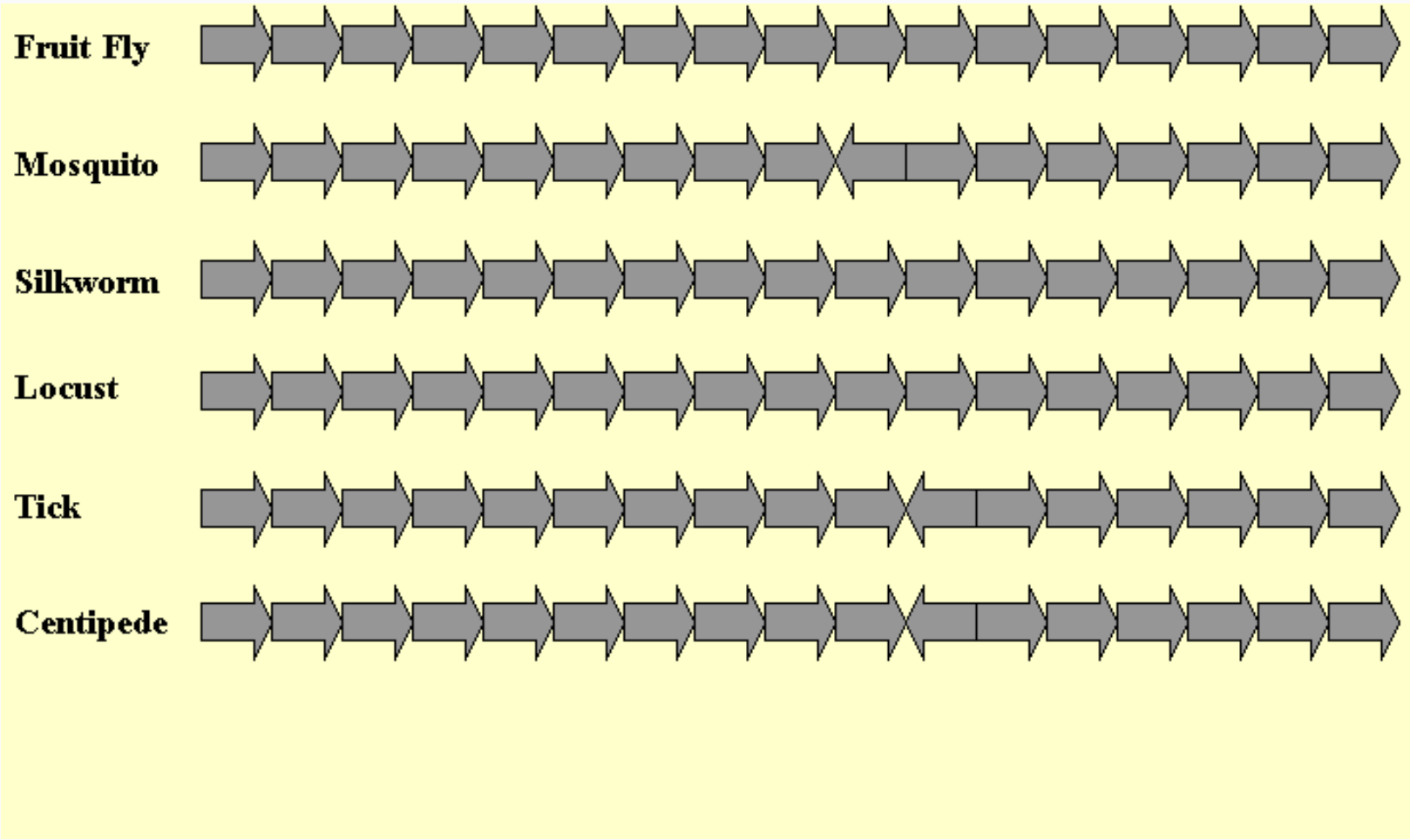
Example:

$$\begin{array}{cccccccccc} G & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ H & = & 0 & 3 & -2 & -1 & 4 & -5 & 6 & 7 \end{array}$$

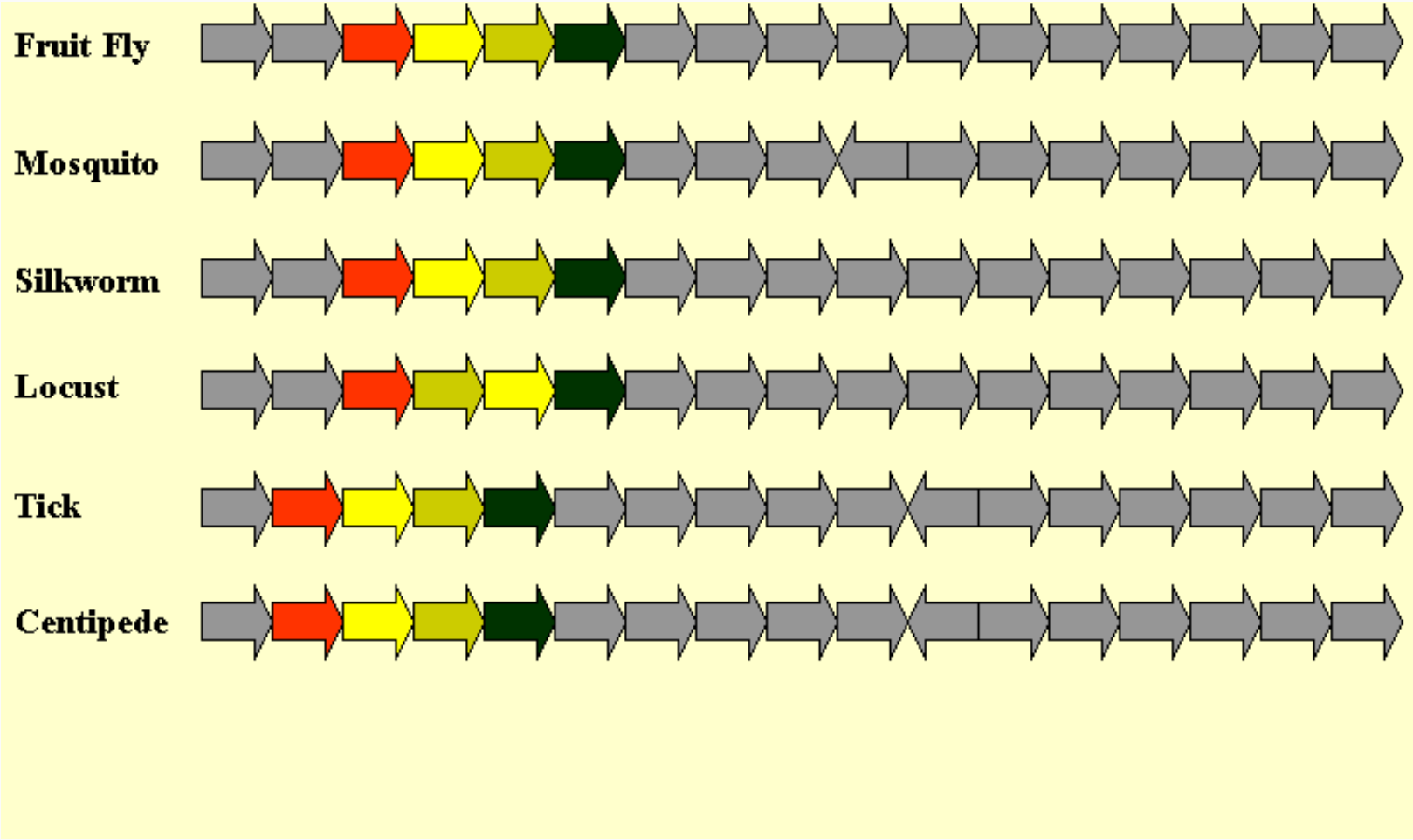
Compact representation ("family portrait"):



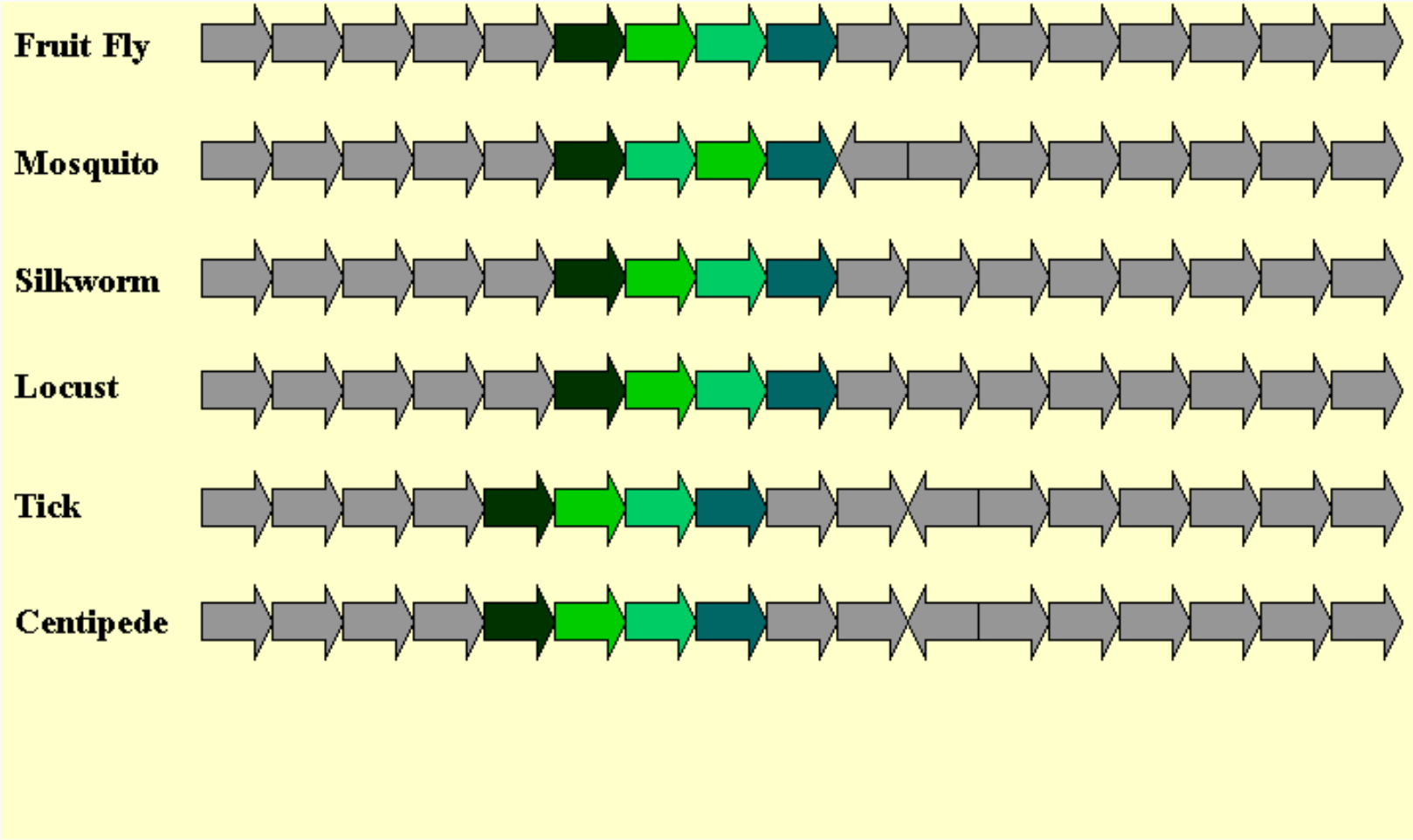
Conserved intervals in mitochondrial genomes of *Arthropoda*



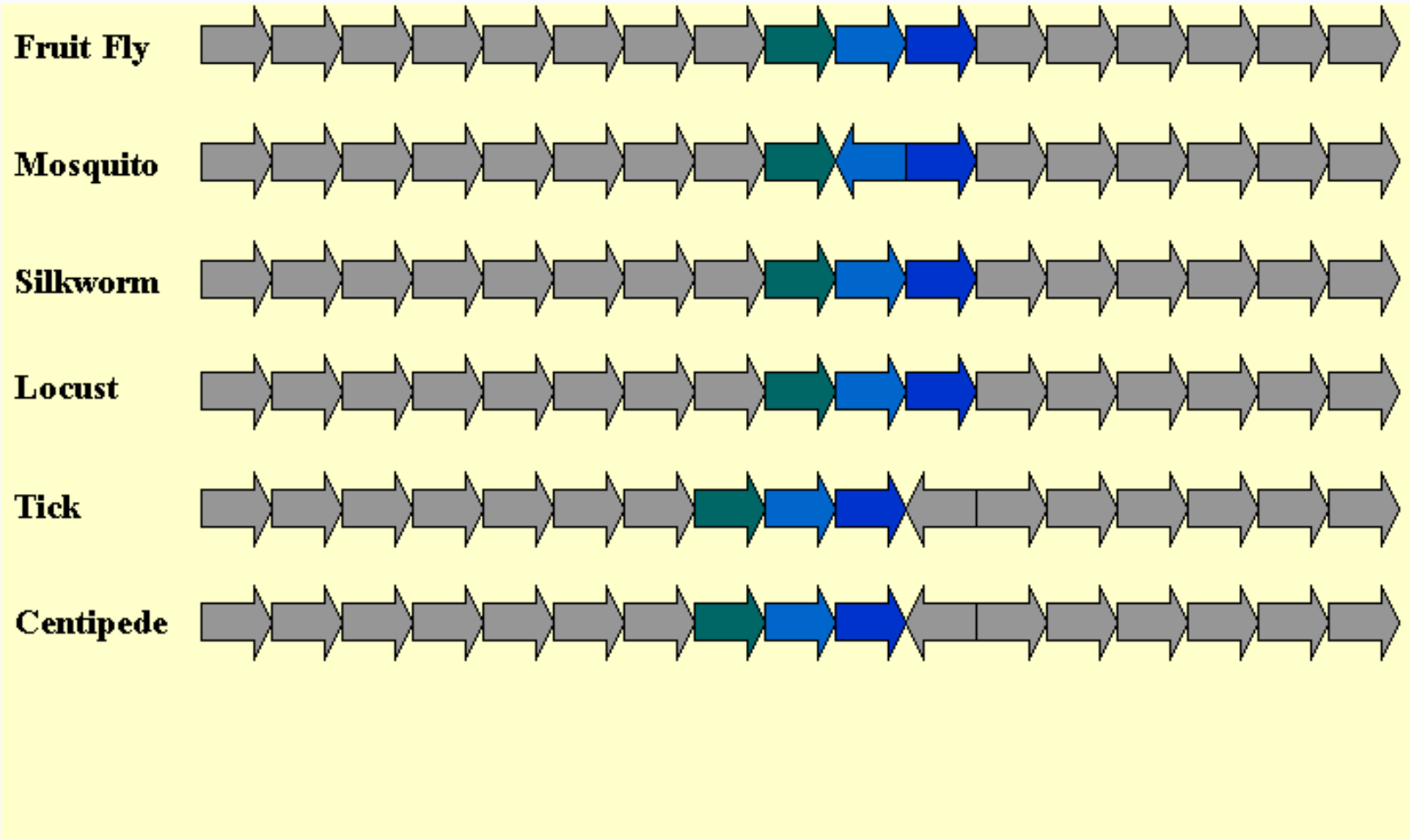
Conserved intervals in mitochondrial genomes of *Arthropoda*



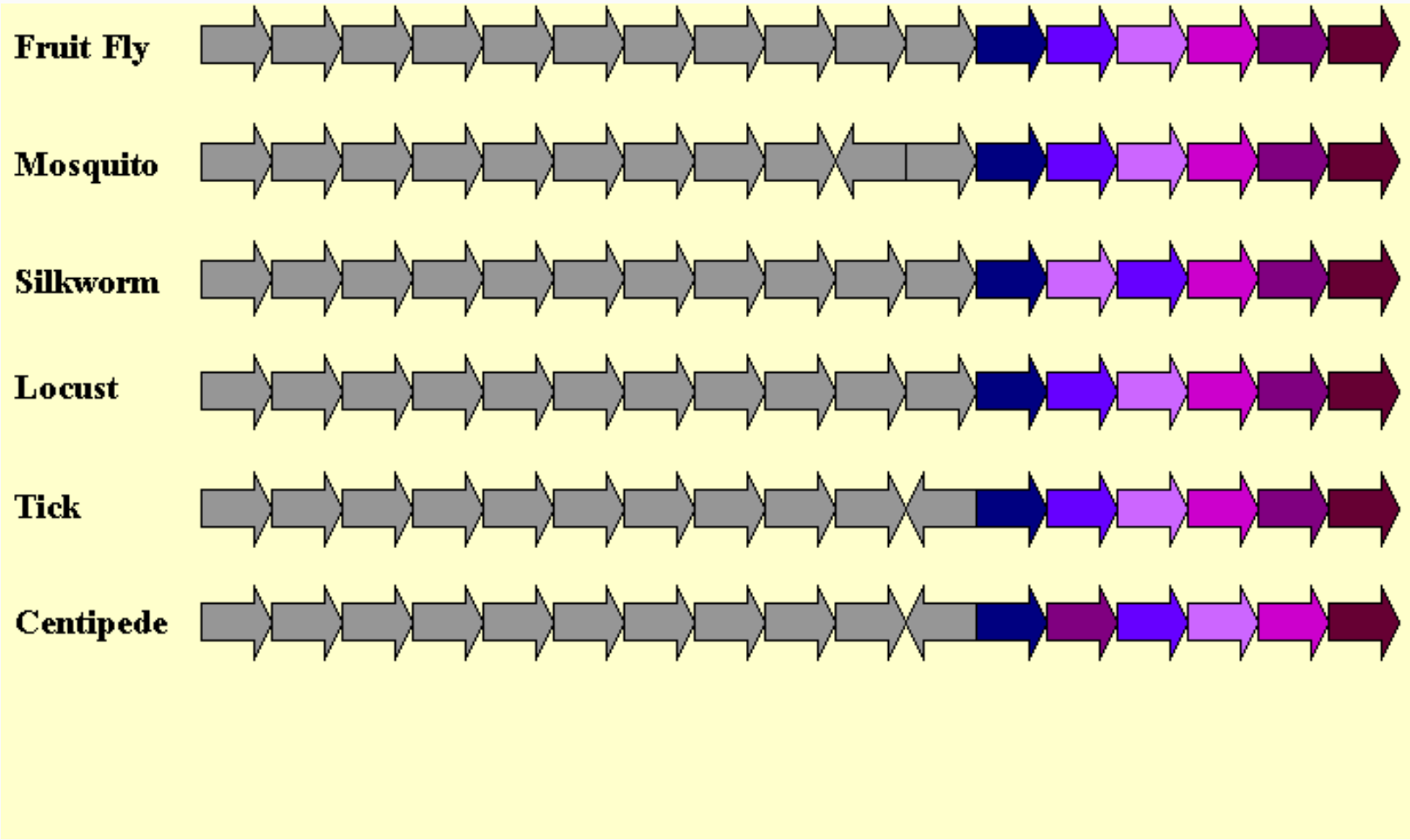
Conserved intervals in mitochondrial genomes of *Arthropoda*



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:

```
stack 0 on S, stack n on M
M_0 ← n
for i = 1, ..., n do
    unstack from M all elements m smaller than |π_i|
    M_i ← m
    stack the element |π_i| on M
    unstack from S all indices s such that (|π_i| < π_s or |π_i| > M_s)
    if i − s = π_i − π_s and M_i = M_s then
        output positive irreducible conserved interval [π_s, π_i]
    end if
    if π_i is positive then
        stack the index i on S
    end if
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

	Fruit Fly		Mosquito		Silkworm		Locust		Tick		Centipede	
Fruit Fly	–	–	90	2	62	1	62	1	158	2	188	3
Mosquito	90	2	–	–	140	3	140	3	200	4	230	5
Silkworm	62	1	140	3	–	–	116	2	180	3	194	4
Locust	62	1	140	3	116	2	–	–	188	3	218	4
Tick	158	2	200	4	180	3	188	3	–	–	110	1
Centipede	188	3	230	5	194	4	218	4	110	1	–	–

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 ot 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:

- Heber & Stoye, Finding all Common Intervals of k Permutations, Proc. CPM 2001.
- Heber & Stoye, Algorithms for Finding Gene Clusters, Proc. WABI 2001.
- Bergeron & Stoye, On the Similarity of Sets of Permutations and its Applications to Genome Comparison, Proc. COCOON 2003.
- Schmidt & Stoye, Quadratic Time Algorithms for Finding Common Intervals in Two and More Sequences, Proc. CPM 2004 (to appear).

Black slide