6.096 Lecture 10

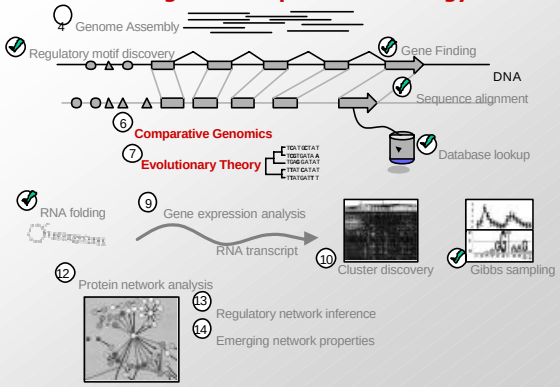
Evolution

Gene correspondence - Rearrangements - Genome duplication

Evolutionary change

Lecture 1 - Introduction
Lecture 2 - Hashing / BLAST
Lecture 3 - Combinatorial Motif Finding
Lecture 4 - Statistical Motif Finding
Lecture 5 - Sequence alignment and Dynamic Programming
Lecture 6 - RNA structure and Context Free Grammars
Lecture 7 - Gene finding and Hidden Markov Models
Lecture 8 - HMMs algorithms and Dynamic Programming
Lecture 9 - Evolutionary change, phylogenetic trees
Lecture 10 - Genome rearrangements, genome duplication

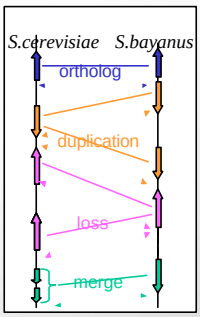
Challenges in Computational Biology



Overview

Genome correspondence
Chromosome evolution
Genome rearrangements
Sorting by reversals
Genome duplication
Duplicate gene evolution
Duplication and rearrangements

Framework: graph of gene correspondence

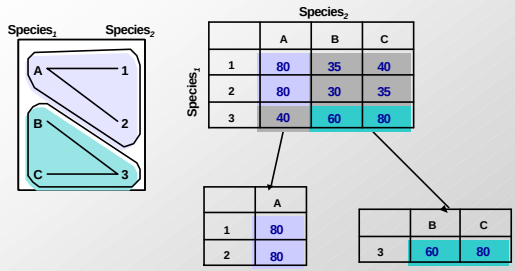


- Weighted bipartite graph
  - Graph represents gene correspondence
  - Nodes: genes (w/ coordinates)
  - Edges: sequence similarity (w/ weights)
- Two types of evolutionary relationships
  - Orthologs (1-to-1 matches)
  - Paralogs (1-to-many / many-to-many)
- Method
  - Eliminate spurious edges (simplify graph)
  - Select edges based on available information
    - Blocks of conserved gene order
    - Protein sequence similarity

Inferring orthologous gene relationships

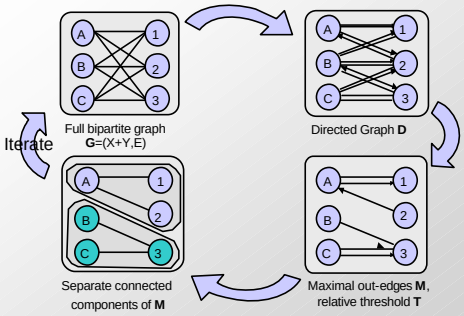
- BBH - Best bi-directional hits
- COG - Clusters of orthologous genes
- BUS - Best unambiguous subgraphs

BUS: Best Unambiguous Subgraphs



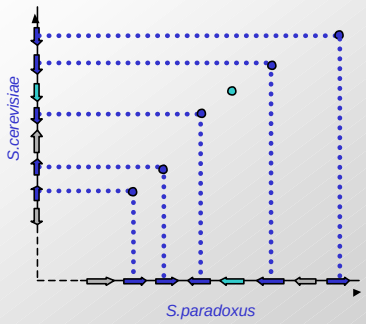
Connected components of all best edges

Implementation: Iterative refinement



Iterative refinement with increasing relative threshold

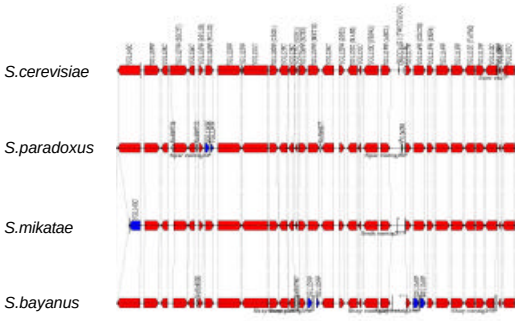
Conservation of gene order (synteny)



Preferentially select edges in synteny blocks

Genomic Dot-Plot (6000 x 6000)

Conservation of local gene order



S.cerevisiae **Chromosome VI 250-300kbp**

Overview

Genome correspondence

Chromosome evolution

Genome rearrangements

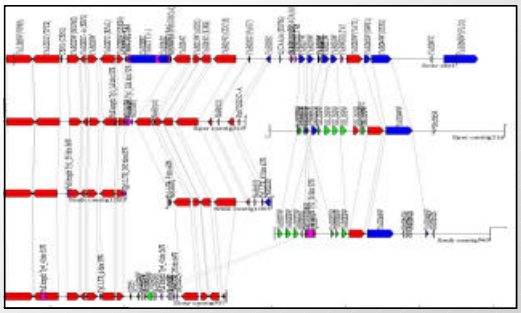
Sorting by reversals

Genome duplication

Duplicate gene evolution

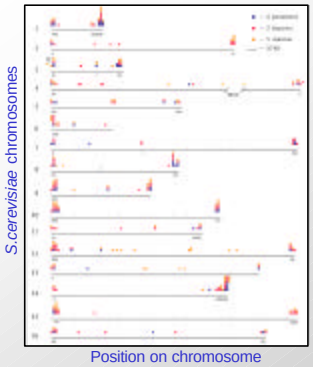
Duplication and rearrangements

Regions of rapid change



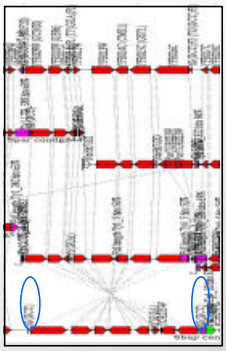
Protein family expansions in chromosome ends

Specific regions of rapid evolution

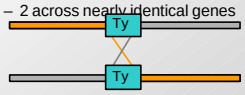


- HXT, FLO, COS, PAU, YRF families
- 80% of ambiguities in 5% of the genome
- 31 of 32 telomeres in ambiguity clusters

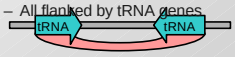
Specific mechanisms mediate rearrangements



- 10 translocations
  - 8 across Ty elements
  - 2 across nearly identical genes



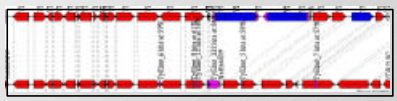
- 19 inversions
  - All flanked by tRNA genes



Evolutionary features

Transposon locations are conserved

- Transposons are active
  - Full-length Ty elements are recent
  - Typically appear in only one genome
- Transposon locations are conserved
  - Recent insertions reuse old loci
  - LTR remnants found in other genomes



Evolutionary advantage of transposons ?

Duhnam, M., et. al. Characteristic Genome Rearrangements in Experimental Evolution of Saccharomyces Cerevisiae. *PNAS* 99, no. 25 (December 10, 2002): 16144-16149. Copyright 2002. National Academy of Sciences, U.S.A.

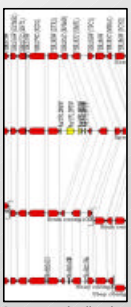
- Studied 8 strains resulting from experimental evolution
  - 3 strains duplicate Chr4R, containing HXT genes
  - 3 strains display extensive overlapping Chr5R deletions
  - 3 strains reuse same breakpoint on Chr14R
- Population advantage in transposon location
  - Ability to mediate reversible rearrangements

Transposons selectively kept in specific loci

Differences in gene content

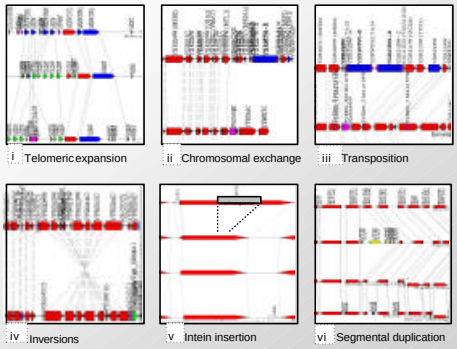
- 8-10 genes unique to each genome
  - Metabolism, regulation/silencing, stress
- Changes in gene dosage
  - 10-20 tandem duplications (1-2 genes)
  - 2 segment duplications (5-6 genes)
- Protein family expansions
  - 211 genes (3%) with ambiguous correspondence
  - Paralog duplication and/or loss

Different species, few novel genes



Segment duplication

Chromosomal Evolution



Overview

Genome correspondence
Chromosome evolution
Genome rearrangements
Sorting by reversals
Genome duplication
Duplicate gene evolution
Duplication and rearrangements

Gene order rearrangement: overlapping inversions

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

1 2 -6 -5 -4 -3 7 8 9 10

1 2 -6 -8 -7 3 4 5 9 10

1 2 -6 -8 -7 3 4 -10 -9 -5

Inference of inversion history: "sorting signed permutations by reversals"

1 2 3 4 5 6 7 8 9 10

1 2 -6 -5 -4 -3 7 8 9 10

1 2 -6 -8 -7 3 4 5 9 10

1 2 -6 -8 -7 3 4 -10 -9 -5

Reversals: Example

π = 1 2 3 4 5 6 7 8

$r(3,5)$

1 2 5 4 3 6 7 8

Reversals: Example

π = 1 2 3 4 5 6 7 8

$r(3,5)$

1 2 5 4 3 6 7 8

$r(5,6)$

1 2 5 4 6 3 7 8

Reversals and Gene Orders

- Gene order is represented by a permutation p:

$p = p_1 \text{ ------ } p_{i-1}\, p_i\, p_{i+1} \text{ ------ } p_{j-1}\, p_j\, p_{j+1} \text{ ----- } p_n$

$r(i,j)$

$p_1 \text{ ------ } p_{i-1}\, p_j\, p_{j-1} \text{ ------ } p_{i+1}\, p_i\, p_{j+1} \text{ ----- } p_n$

- Reversal r (i, j) reverses (flips) the elements from i to j in p

Reversal Distance Problem

- Goal: Given two permutations, find the shortest series of reversals that transforms one into another
- Input: Permutations p and s
- Output: A series of reversals $r_1,\ldots r_t$ transforming p into s, such that t is minimum
- t - reversal distance between p and s
- $d(p, \sigma)$ - smallest possible value of t, given p and σ

Sorting By Reversals Problem

- Goal: Given a permutation, find a shortest series of reversals that transforms it into the identity permutation ($1\ 2\ \ldots\ n$)
- Input: Permutation p
- Output: A series of reversals $r_1, \ldots r_t$ transforming p into the identity permutation such that t is minimum

Sorting By Reversals: Example

- $t = d(p)$ - reversal distance of p
- Example :

p = 3 4 2 1 5 6 7 10 9 8
4 3 2 1 5 6 7 10 9 8
4 3 2 1 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10

So $d(p) = 3$

Sorting by reversals: 5 steps

Step 0: p	2	-4	-3	5	-8	-7	-6	1
Step 1:	2	3	4	5	-8	-7	-6	1
Step 2:	2	3	4	5	6	7	8	1
Step 3:	2	3	4	5	6	7	8	-1
Step 4:	-8	-7	-6	-5	-4	-3	-2	-1
Step 5: g	1	2	3	4	5	6	7	8

Sorting by reversals: 4 steps

Step 0: p	2	-4	-3	5	-8	-7	-6	1
Step 1:	2	3	4	5	-8	-7	-6	1
Step 2:	-5	-4	-3	-2	-8	-7	-6	1
Step 3:	-5	-4	-3	-2	-1	6	7	8
Step 4: g	1	2	3	4	5	6	7	8

Sorting by reversals: 4 steps

Step 0: p	2	-4	-3	5	-8	-7	-6	1
Step 1:	2	3	4	5	-8	-7	-6	1
Step 2:	-5	-4	-3	-2	-8	-7	-6	1
Step 3:	-5	-4	-3	-2	-1	6	7	8
Step 4: g	1	2	3	4	5	6	7	8

What is the reversal distance for this permutation? Can it be sorted in 3 steps?

Pancake Flipping Problem

- The chef is sloppy; he prepares an unordered stack of pancakes of different sizes
- The waiter wants to rearrange them (so that the smallest winds up on top, and so on, down to the largest at the bottom)
- He does it by flipping over several from the top, repeating this as many times as necessary

Christos Papadimitrou and Bill Gates flip pancakes

Pancake Flipping Problem: Formulation

- Goal: Given a stack of *n* pancakes, what is the minimum number of flips to rearrange them into perfect stack?
- Input: Permutation *p*
- Output: A series of prefix reversals $r_1, \ldots r_t$ transforming *p* into the identity permutation such that *t* is minimum

Pancake Flipping Problem: Greedy Algorithm

- Greedy approach: 2 prefix reversals at most to place a pancake in its right position, *2n – 2* steps total

  at most
- William Gates and Christos Papadimitriou showed in the mid-1970s that this problem can be solved by at most *5/3 (n + 1) prefix reversals*

Sorting By Reversals: A Greedy Algorithm

- If sorting permutation *p* = 1 2 3 6 4 5, the first three elements are already in order so it does not make any sense to break them.
- The length of the already sorted prefix of *p* is denoted *prefix*(*p*)
  - *prefix*(*p*) = 3
- This results in an idea for a greedy algorithm: increase *prefix*(*p*) at every step

Greedy Algorithm: An Example

- Doing so, *p* can be sorted

1 2 3 6 4 5

1 2 3 4 6 5

1 2 3 4 5 6

- Number of steps to sort permutation of length *n* is at most *(n – 1)*

Greedy Algorithm: Pseudocode

```
SimpleReversalSort(p)
1 for  i ← 1 to n – 1
2   j ← position of element i in p (i.e., p_j = i)
3   if  j ? i
4     p ← p * r(i, j)
5     output p
6   if p is the identity permutation
7     return
```

Analyzing SimpleReversalSort

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $p = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 1: 1 6 2 3 4 5
  - Step 2: 1 2 6 3 4 5
  - Step 3: 1 2 3 6 4 5
  - Step 4: 1 2 3 4 6 5
  - Step 5: 1 2 3 4 5 6

Analyzing SimpleReversalSort (cont'd)

- But it can be sorted in two steps:

  p = 6 1 2 3 4 5
  - Step 1: 5 4 3 2 1 6
  - Step 2: 1 2 3 4 5 6
- So, SimpleReversalSort(p) is not optimal
- Optimal algorithms are unknown for many problems; approximation algorithms are used

Approximation Algorithms

- These algorithms find approximate solutions rather than optimal solutions
- The approximation ratio of an algorithm A on input p is:

$$A(p) / OPT(p)$$

where

A(p) -solution produced by algorithm A
OPT(p) - optimal solution of the problem

Approximation Ratio/Performance Guarantee

- Approximation ratio (performance guarantee) of algorithm A: max approximation ratio of all inputs of size n
  - For algorithm A that minimizes objective function (minimization algorithm):
    - $\max_{|p| = n} A(p) / OPT(p)$

Approximation Ratio/Performance Guarantee

- Approximation ratio (performance guarantee) of algorithm A: max approximation ratio of all inputs of size n
  - For algorithm A that minimizes objective function (minimization algorithm):
    - $\max_{|p| = n} A(p) / OPT(p)$
  - For maximization algorithm:
    - $\min_{|p| = n} A(p) / OPT(p)$

Adjacencies and Breakpoints

$p = p_1 p_2 p_3 \cdots p_{n-1} p_n$

- A pair of elements p_i and p_{i+1} are adjacent if
  $p_{i+1} = p_i \pm 1$
- For example:
  $p = 1\ 9\ 3\ 4\ 7\ 8\ 2\ 6\ 5$
- (3, 4) or (7, 8) and (6,5) are adjacent pairs

Breakpoints: An Example

There is a breakpoint between any pair of non-adjacent elements:

$p = 1\ 9\ |\ 3\ 4\ |\ 7\ 8\ |\ 2\ 6\ |\ 5\ |$

- Pairs (1,9), (9,3), (4,7), (8,2) and (2,5) form breakpoints of permutation p
- $b(p)$ - # breakpoints in permutation p

Extending Permutations

- We put two elements $p_0 = 0$ and $p_{n+1} = n+1$ at the ends of p

Example:

$\pi = 1\ |\ 9\ |\ 3\ 4\ |\ 7\ 8\ |\ 2\ |\ 6\ 5$

Extending with 0 and 10

$\pi = 0\ 1\ |\ 9\ |\ 3\ 4\ |\ 7\ 8\ |\ 2\ |\ 6\ 5\ |\ 10$

Note: A new breakpoint was created after extending

Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.

$p = 2\ 3\ 1\ 4\ 6\ 5$

$0\ 2\ 3\ 1\ 4\ 6\ 5\ 7$	$b(p) = 5$
$0\ 1\ 3\ 2\ 4\ 6\ 5\ 7$	$b(p) = 4$
$0\ 1\ 2\ 3\ 4\ 6\ 5\ 7$	$b(p) = 2$
$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7$	$b(p) = 0$

Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.
- This implies:

reversal distance = #breakpoints / 2

$p = 2\ 3\ 1\ 4\ 6\ 5$

$0\ 2\ 3\ 1\ 4\ 6\ 5\ 7$	$b(p) = 5$
$0\ 1\ 3\ 2\ 4\ 6\ 5\ 7$	$b(p) = 4$
$0\ 1\ 2\ 3\ 4\ 6\ 5\ 7$	$b(p) = 2$
$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7$	$b(p) = 0$

Sorting By Reversals: A Better Greedy Algorithm

BreakPointReversalSort(p)

1 while $b(p) > 0$
2 Among all possible reversals, choose reversal r minimizing $b(p \cdot r)$
3 $p \leftarrow p \cdot r(i, j)$
4 output p
5 return

Sorting By Reversals: A Better Greedy Algorithm

BreakPointReversalSort(p)

1 while b(p) > 0
2 Among all possible reversals, choose reversal r minimizing b($p \cdot r$)
3 $p \leftarrow p \cdot r$(i, j)
4 output p
5 return

Problem: this algorithm may work forever

Strips

- Strip: an interval between two consecutive breakpoints in a permutation
  - Decreasing strip: *strip* of elements in decreasing order (e.g. 6 5 and 3 2).
  - Increasing strip: *strip* of elements in increasing order (e.g. 7 8)

0 1 9 4 3 7 8 2 5 6 *10*

- A single-element strip can be declared either increasing or decreasing. We will choose to declare them as decreasing with exception of the strips with *0* and *n+1*

Reducing the Number of Breakpoints

Theorem 1:

If permutation p contains at least one decreasing strip, then there exists a reversal r which decreases the number of breakpoints (i.e. $b(p \cdot r) < b(p)$)

Things To Consider

- For p = 1 4 6 5 7 8 3 2
  0 1 4 6 5 7 8 3 2 *9* $b(p) = 5$
  - Choose decreasing strip with the smallest element k in p (k = 2 in this case)

Things To Consider (cont'd)

- For p = 1 4 6 5 7 8 3 2
  0 1 4 6 5 7 8 3 2 *9* $b(p) = 5$
  - Choose decreasing strip with the smallest element k in p (k = 2 in this case)

Things To Consider (cont'd)

- For p = 1 4 6 5 7 8 3 2
  0 1 4 6 5 7 8 3 2 *9* $b(p) = 5$
  - Choose decreasing strip with the smallest element k in p (k = 2 in this case)
  - Find $k - 1$ in the permutation

Things To Consider (cont'd)

- For p = 1 4 6 5 7 8 3 2
  0 1 4 6 5 7 8 3 2 9 $b(p) = 5$
  - Choose decreasing strip with the smallest element k in p (k = 2 in this case)
  - Find $k - 1$ in the permutation
  - Reverse the segment between k and k-1:
  - 0 1 4 6 5 7 8 3 2 9 $b(p) = 5$
  - 0 1 2 3 8 7 5 6 4 9 $b(p) = 4$

Reducing the Number of Breakpoints Again

- If there is no decreasing strip, there may be no reversal r that reduces the number of breakpoints (i.e. $b(p \bullet r) = b(p)$ for any reversal r).
- By reversing an increasing strip (# of breakpoints stay unchanged), we will create a decreasing strip at the next step. Then the number of breakpoints will be reduced in the next step (theorem 1).

Things To Consider (cont'd)

- There are no decreasing strips in p, for:

p = 0 1 2 5 6 7 3 4 8 $b(p) = 3$

$p \bullet r(6,7)$ = 0 1 2 5 6 7 4 3 8 $b(\pi) = 3$

- ✓ $r(6,7)$ does not change the # of breakpoints
- ✓ $r(6,7)$ creates a decreasing strip thus guaranteeing that the next step will decrease the # of breakpoints.

ImprovedBreakpointReversalSort

```
ImprovedBreakpointReversalSort(p)
1 while b(p) > 0
2    if p has a decreasing strip
3       Among all possible reversals, choose reversal r
                    that minimizes b(p • r)
4    else
5       Choose a reversal r that flips an increasing strip in p
6    p ← p • r
7    output p
8 return
```

ImprovedBreakpointReversalSort: Performance Guarantee

- *ImprovedBreakPointReversalSort* is an approximation algorithm with a performance guarantee of at most 4
  - It eliminates at least one breakpoint in every two steps; at most $2b(p)$ steps
  - Approximation ratio: $2b(p) / d(p)$
  - Optimal algorithm eliminates at most 2 breakpoints in every step: $d(p) \geq b(p) / 2$
  - Performance guarantee:
    - $(2b(p) / d(p)) \leq [2b(p) / (b(p) / 2)] = 4$

Signed Permutations

- Up to this point, all permutations to sort were unsigned
- But genes have directions… so we should consider signed permutations

p = 1 -2 -3 4 -5

GRIMM Web Server

- Real genome architectures are represented by signed permutations
- Efficient algorithms to sort signed permutations have been developed
- GRIMM web server computes the reversal distances between signed permutations:
  http://nbcr.sdsc.edu/GRIMM/grimm.cgi

GRIMM Web Server

http://www-cse.ucsd.edu/groups/bioinformatics/GRIMM

Courtesy of Glenn Tesler. Used with permission.

Sorting by reversals

Bader DA, Moret BM, Yan M. (2001) A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J* Comput Biol 8:483-91.

Bergeron A. [in press] A very elementary presentation of the Hannenhalli-Pevzner theory. Discrete Applied Mathematics.

Hannenhalli, S. (1996). Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71:137–151.

Hannenhalli, S. and Pevzner, P. (1995). Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, pages 581–592.

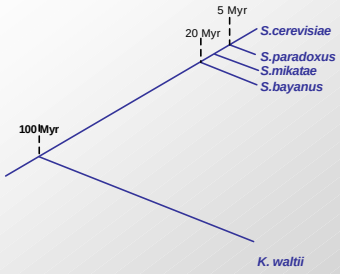
Hannenhalli, S. and Pevzner, P. A. (1999). Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Journal of the ACM*, 48:1–27.

Tesler G (2002) GRIMM: genome rearrangements web server. Bioinformatics, 18:492-3.

Overview

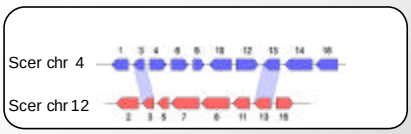
Genome correspondence
Chromosome evolution
Genome rearrangements
Sorting by reversals
Genome duplication
Duplicate gene evolution
Duplication and rearrangements

Further back in evolutionary time



Ability to ask different set of questions

Whole Genome Duplication (WGD) in Yeast?



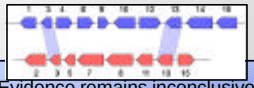
Wolfe '97

- Genomic evidence
  - Conserved order of paralogous genes
  - Same transcriptional orientation
- However
  - Interspersed with single-copy genes

Interpretation: Genome duplication followed by gene loss

Whole genome duplication is controversial

- Insufficient evidence
  - Only 50% of genome in duplicate regions
  - Only 8% of genes present in two copies
  - Extensive redundancy outside duplicate regions
- Evidence against WGD
  - Divergence-based dating show multiple times
  - Other species have similar level of redundancy
- Alternative evolutionary scenario proposed
  - Independent segmental duplications
  - Also consistent with the evidence



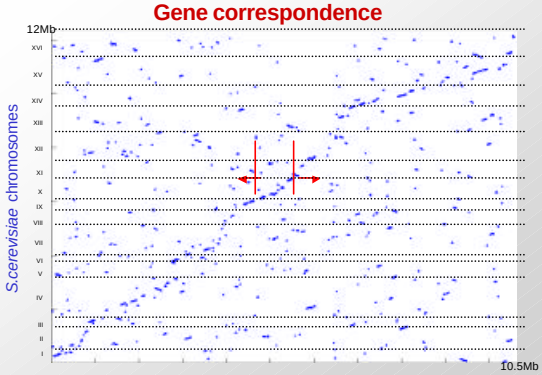
Evidence remains inconclusive

- "There was a whole-genome duplication." **Wolfe, *Nature* '97**
- "There was no whole-genome duplication." **Dujon, *FEBS* 2000**
- "At least some chrom dup. occurred independently" **Langkjaer, *JMB*, 2000**
- "Dynamic equilibrium of duplications and loss" **Llorente, *FEBS*, 2000**
- "Recent evidence supports single event". **Wong, *PNAS* '02**
- "Continuous block duplications and deletions" **Dujon, *Yeast* 2003**
- "Dup. precedes divergence from Kluyveromyces." **Piskur, *Nature*, 2003**
- "Telomere-mediated duplication events" **Coissac, Mol Bio Evo 1997**
- "Multiple closely spaced events" **Friedman, Genome Res, 2003**
- "Spontaneous duplication of large chromosomal segments" **Koszul, EMBO '04**

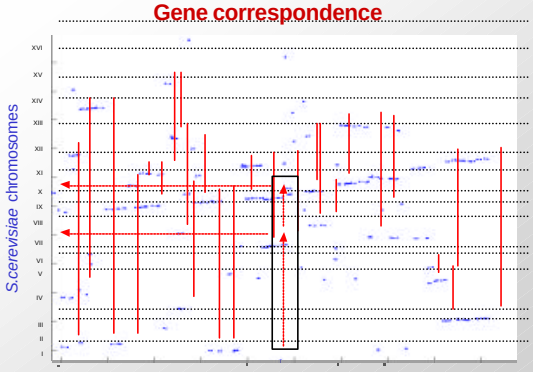
Missing evidence supporting WGD

Non-duplicated relative

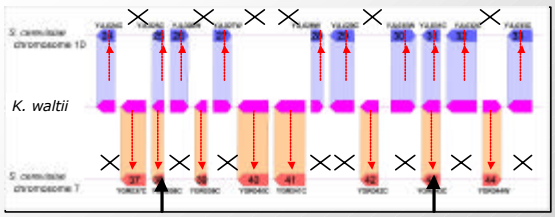
Gene correspondence



Gene correspondence



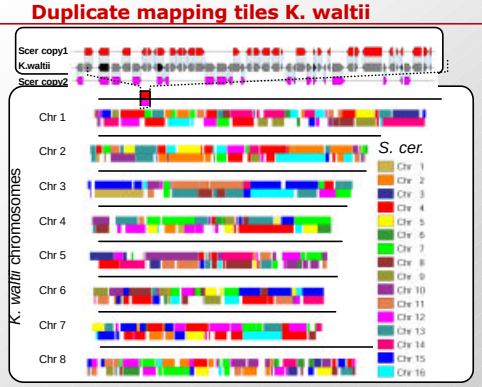
Sister regions show gene interleaving



Few genes remain in 2 copies

Gene interleaving is evidence of complete duplication

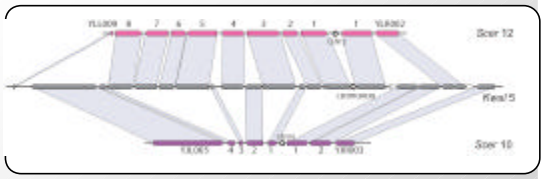
Duplicate mapping tiles K. waltii



Doubly Conserved Synteny Blocks (DCS)

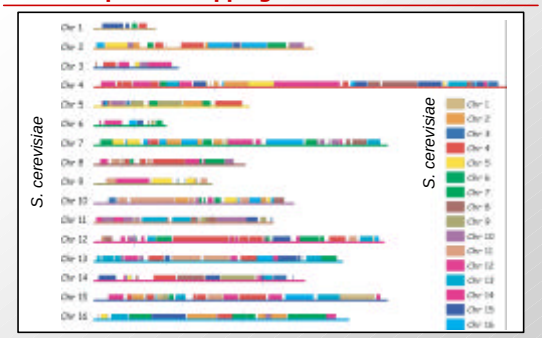
- 253 DCS blocks were identified containing 75% of K. waltii genes and 81% of S. cerevisae genes
- A typical DCS block has 27 genes (largest block has 81 genes).
- DCS blocks are separated by ~3 genes on the average.
- In a DCS block 90% of Kw genes have a match in at least 1 of the 2 Sc regions.
- 47 blocks have no duplicated gene.

Duplicate mapping of centromeres



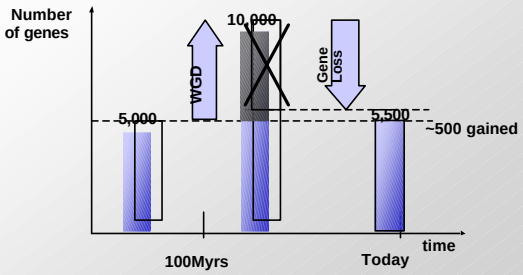
Recognize sister regions solely based on gene order

Duplicate mapping tiles *S. cerevisiae*



145 blocks cover 88% of genome

Whole-genome duplication resolved



Overview

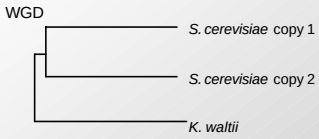
Genome correspondence
Chromosome evolution
Genome rearrangements
Sorting by reversals
Genome duplication
Duplicate gene evolution
Duplication and rearrangements

Accelerated gene divergence

- Ohno hypothesized that after duplication, one copy would preserve the original function, and the other copy would be free to diverge. Others argued that both copies would diverge.
- 76 of 457 duplicated gene pairs show accelerated evolution. In 95% of the cases, acceleration was limited to one of the 2 paralogs.
- Deletion of the ancestral paralog is lethal in 18% of the cases.
- Deletion of a derived paralog is never lethal.

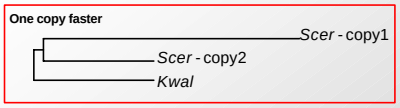
Fate of duplicated genes

- 457 genes kept in two copies, result of selection
  - Involved in sugar metabolism and fermentation

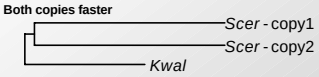


Evidence of accelerated protein divergence ?

Scenarios for rapid gene evolution



Ohno, 1970

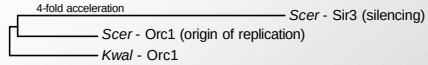


Lynch, 2000

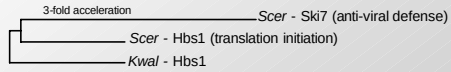
20% of duplicated genes show acceleration
95% of cases: Only one copy faster

Emerging gene functions after duplication

- Origin of replication → silencing



- Translation initiation → anti-viral defense



Asymmetric divergence → recognize ancestral / derived

Distinct functional properties

	Ancestral function	Derived function
Gene deletion	Lethal (20%)	Never lethal

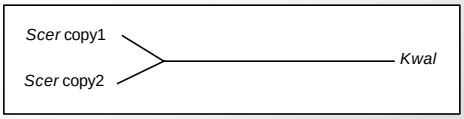
Gain new function and lose ancestral function

Distinct functional properties

	Ancestral function	Derived function
Gene deletion	Lethal (20%)	Never lethal
Expression	Abundant	Specific (stress, starvation)
Localization	General	Specific (mitochondrion, spores)

Gain new function and lose ancestral function

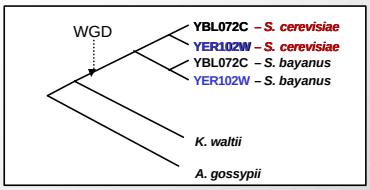
Decelerated evolution



- 60 gene pairs (13% of 457 pairs)
  - 98% protein identity (all pairs: 55%)
  - 90% identity in 4fold degenerate sites (all pairs: 41%)
- Not recent duplication
  - Gene order argues ancestral WGD pairs

Gene conversion?

Evidence of gene conversion

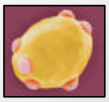


- Tree root reveals time of duplication
  - No acceleration in the *K. waltii* branch
  - The two genes have recently replaced each other
- Branching order reveals gene conversion
  - Paralogs are closer to each other than to their ortholog
  - Both *S. cerevisiae* and *S. bayanus* show gene conversion

Periodic gene conversion

Evolutionary genomics in yeast

- Genome ancestry resolved
  - Whole -genome duplication
  - Massive gene loss

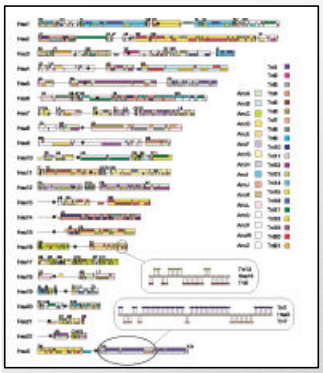


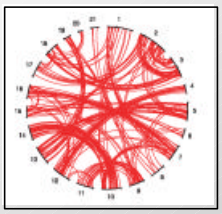
- Emergence of new functions
  - Asymmetric acceleration
  - Ancestral and derived functions
  - Repository for buffering mutations

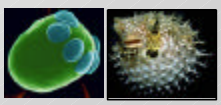
Overview

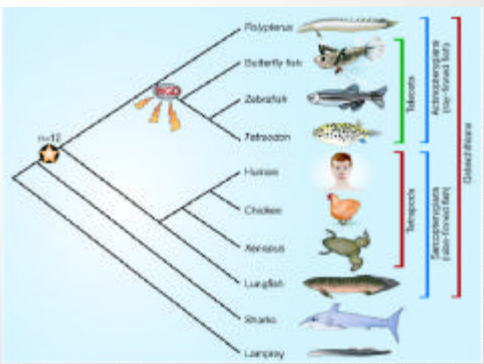
Genome correspondence
Chromosome evolution
Genome rearrangements
Sorting by reversals
Genome duplication
Duplicate gene evolution
Duplication and rearrangements

Genome duplication in a vertebrate

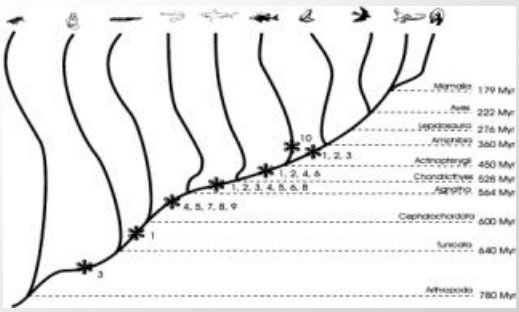








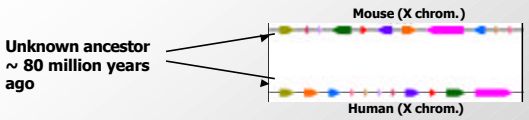
Mammals: How many WGDs?



How did the pre-duplicated ancestor look like?

- Can we derive the architecture of the current (human and tetraodon genomes) genomes in terms of the common ancestor?
- What was the sequence of rearrangement events after WGD?

Genome rearrangements

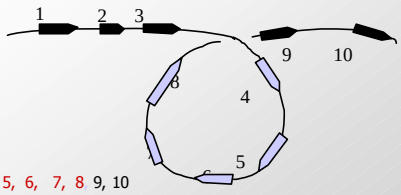


- What is the architecture of the ancestral genome?
- What is the evolutionary scenario for transforming one genome into the other?

History of Chromosome X

Rat Consortium, *Nature*, 2004

Reversals

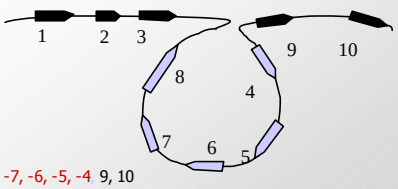


1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.
- In the course of evolution, blocks 1, 2, ...9, 10 could be transformed into

1, 2, 3, -8, -7, -6, -5, -4, 9, 10.

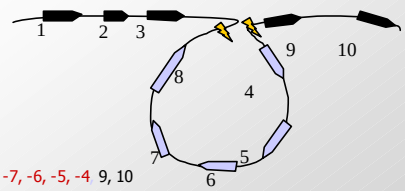
Reversals



1, 2, 3, -8, -7, -6, -5, -4, 9, 10

- Blocks represent conserved genes.
- In the course of evolution, blocks 1,…,10 could be misread as
  1, 2, 3, -8, -7, -6, -5, -4, 9, 10.
- **Evolution:** occurred one-two times every million years on the evolutionary path between human and mouse.

Reversals



1, 2, 3, -8, -7, -6, -5, -4, 9, 10

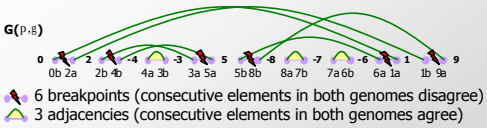
The inversion introduced two breakpoints (disruptions in order).

Sorting by reversals
Most parsimonious scenarios

Step 0: p 2 -4 -3 5 -8 -7 -6 1
Step 1: 2 3 4 5 -8 -7 -6 1
Step 2: -5 -4 -3 -2 -8 -7 -6 1
Step 3: -5 -4 -3 -2 -1 6 7 8
Step 4: g 1 2 3 4 5 6 7 8

The **reversal distance** is the minimum number of reversals required to transform π into γ.
Here, the reversal distance is d=4.

Breakpoint graph



6 breakpoints (consecutive elements in both genomes disagree)
3 adjacencies (consecutive elements in both genomes agree)

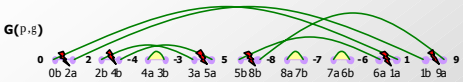
- **DualityTheorem:**

reversal distance = #genes + 1 – #cycles + h

where h is rather complicated, but can be computed from breakpoint graph in polynomial time.

- Here, reversal distance = 8 + 1 – 5 + 0 + 0 = 4

Breakpoint graph

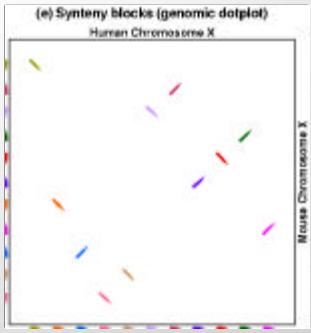


6 breakpoints (consecutive elements in both genomes disagree)
3 adjacencies (consecutive elements in both genomes agree)

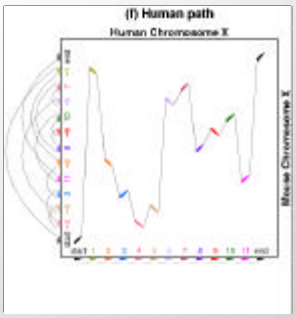
- **Duality Theorem for Sorting by Reversals - simple and imprecise version.**

reversal distance = number of elements + 1 – number of cycles

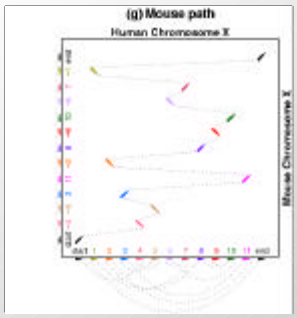
Constructing Breakpoint Graph: Dot Plot



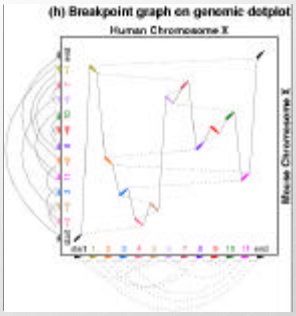
Constructing Breakpoint Graph: Black Path



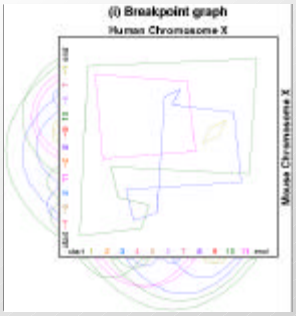
Constructing Breakpoint Graph: Gray Path



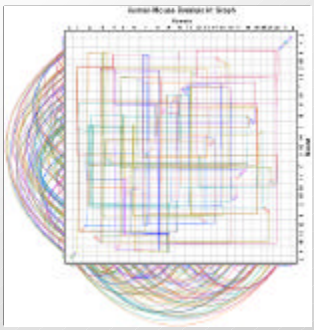
Constructing Breakpoint Graph: Superimposing Two Paths



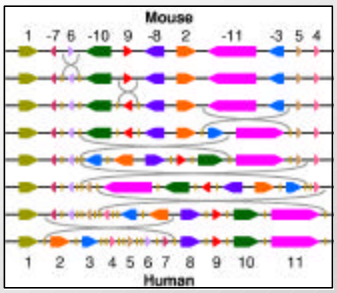
Constructing Breakpoint Graph: Removing Dot-Plot



Human-mouse breakpoint graph



Constructing Rearrangement Scenarios



Reconstructing pre-duplicated Genome

- WGD of genome *R* results in perfect duplicated genome *R+R*
- *R+R* becomes subject to rearrangements that shuffle genes in *R+R* and result in some rearranged duplicated genome *P*
- Problem: reconstruct pre-duplicated genome *R* from rearranged duplicated genome *P*.

Genome Halving Problem

- WGD of genome *R* results in perfect duplicated genome *R+R*
- *R+R* becomes subject to rearrangements that shuffle genes in *R+R* and result in some rearranged duplicated genome *P*
- Problem: reconstruct pre-duplicated genome *R* from rearranged duplicated genome *P*.
- Genome Halving Problem: Given a duplicated genome *P*, recover the ancestral pre-duplicated genome *R* minimizing the reversal distance from *R+R* to *P*

Illustration

R=+a -d +e -c +b

R+R=+a -d +e -c +b +a -d +e -c +b
+a -d +d -a -b +c -e +e -c +b
+a -d +d +c -e +e -c -c +b +a +b
+a -d +d -e +e -c -c +b +a +b

P=+a -d +e -d +e -c -c +b +a +b

Recover it!

R = ?? ?? ?? ?? ??

? ? ?

P = +a -d +e -d +e -c -c +b +a +b

Even worse!

R = ?? ?? ?? ?? ??

? ? ?

P = +a -d +e -d +e -c -c +b +a +b

Suppose we somehow figured out what is *R*: would it help to find d(P,R+R)?

R = +a +c +d +e +b

R+R = +a +c +d +e +b +a +c +d +e +b

P = +a -d +e -d +e -c -c +b +a +b

???

+a +c +d +e +b

+a +c +d +e +b +a +c +d +e +b
+a -e -d -c -a -b -e -d -c +b
+a -e -d -c +c +d +e +b +a +b
+a -e -d -e -d -c +c +b +a +b
+a +d +e +d +e -c +c +b +a +b
+a +d +e +d +e -c -c +b +a +b
+a +d +e -d +e -c -c +b +a +b

+a -d +e -d +e -c -c +b +a +b

HP-theory: reminder

- Transforming signed gene order
  +a +b –c
  into unsigned gene order
  $a^t a^h\ b^t b^h\ c^h c^t$
- Elements x^t and x^h are called **obverse pair**
- *t* stands for ***tail*** and *h* stands for ***head***

Breakpoint graph is formed by 3 matchings:

obverse matching
black matching (adjacent elements in 1st permutation)
gray matching (adjacent elements in 2nd permutation)

HP-theory: reminder

- Breakpoint graph is formed by obverse, black and gray matchings.
- Every pair of matching forms a collection of alternating cycles:

  black-gray cycles (#cycles in HP theory)
  single black-obverse cycle (1st permutation)
  single gray-obverse cycle (2nd permutation)

reversal distance between two circular permutations =

#elements - #black-gray cycles

Reversal distance between duplicated genomes

- While there exist fast algorithms for computing reversal distance between permutations (i.e., no duplicate genes), the problem of computing reversal distance between genomes with duplicated genes remains unsolved.
- Solution: label different copies of each gene (*k!* different labelings for a gene with *k* copies)
- One of these labelings is unavoidably an

  optimal labeling

  corresponding to the optimum rearrangement scenario
- Running time: $(k!)^n$ invocations of HP algorithms for a genome with *n* genes each present in *k* copies.

Labelings and breakpoint graphs

- Every labeling transforms genomes with duplicated genes into genomes without duplicated genes and enables applications of HP algorithm.
- Every labeling corresponds to a breakpoint graph
- Good labelings correspond to breakpoint graphs with large number of cycles.
- Can we construct a labeling corresponding to a large number of cycles?

Rearrangements in Duplicated Genomes: Challenges.

- Computing *d(P,Q).* Can we construct a labeling of duplicated genomes *P* and *Q* maximizing the number of cycles? NO

Rearrangements in Duplicated Genomes: Challenges.

- Computing *d(P,Q).* Can we construct a labeling of duplicated genomes *P* and *Q* maximizing the number of cycles? NO
- Computing *d(P,R+R).* Can we construct a labeling of duplicated genomes *P* and *R+R* maximizing the number of cycles? NO

Rearrangements in Duplicated Genomes: Challenges.

- Computing *d(P,Q).* Can we construct a labeling of duplicated genomes *P* and *Q* maximizing the number of cycles? NO
- Computing *d(P,R+R).* Can we construct a labeling of duplicated genomes *P* and *R+R* maximizing the number of cycles? NO
- Computing $\min_R d(P,R+R)$.

Rearrangements in Duplicated Genomes: Challenges.

- Computing *d(P,Q).* Can we construct a labeling of duplicated genomes *P* and *Q* maximizing the number of cycles? NO
- Computing *d(P,R+R).* Can we construct a labeling of duplicated genomes *P* and *R+R* maximizing the number of cycles? NO
- Computing $\min_R d(P,R+R)$. ***YES!***

Rearrangements in Duplicated Genomes: Challenges.

- Breakpoint graphs are not defined for duplicated genomes.
- Can we generalize the notion of breakpoint graph for the case of duplicated genomes?
- Idea: Explore the connection between de Bruijn graphs and breakpoint graphs.

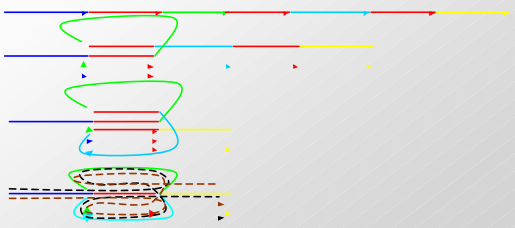
De Bruijn Graphs

- De Bruijn graph: Given a set of edge-labeled graphs, de Bruijn graph of this set is the result of "gluing" edges with the same label in all graphs in the set.

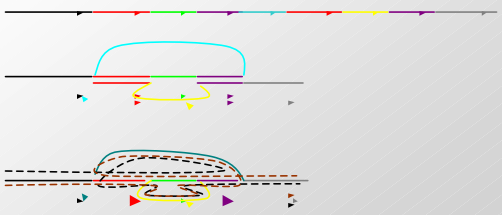
De Bruijn Graphs

- De Bruijn graph: Given a set of edge-labeled graphs, de Bruijn graph of this set is the result of "gluing" edges with the same label in all graphs in the set.
- Did we see de Bruijn graphs today?

De Bruijn graph of a path



de Bruijn graph of another path



De Bruijn Graphs

- De Bruijn graph: Given a set of edge-labeled graphs, de Bruijn graph of this set is the result of "gluing" edges with the same label in all graphs in the set.
- Did we see de Bruijn graphs today?

De Bruijn Graphs

- De Bruijn graph: Given a set of edge-labeled graphs, de Bruijn graph of this set is the result of "gluing" edges with the same label in all graphs in the set.
- Did we see de Bruijn graphs today?
- Breakpoint graph of permutations *P* and *Q*
  ==
  de Bruijn graph of *P*-cycle and *Q*-cycle

De Bruijn Graphs

- Breakpoint graph of permutations *P* and *Q*
  ==
  de Bruijn graph of *P*-cycle and *Q*-cycle
- Breakpoint graph of any genomes *P* and *Q*
  (with multiple gene copies)

De Bruijn Graphs

- Breakpoint graph of permutations *P* and *Q*
  ==
  de Bruijn graph of *P*-cycle and *Q*-cycle
- Breakpoint graph of any genomes *P* and *Q*
  (with multiple gene copies)
  ==
  de Bruijn graph of *P*-cycle and *Q*-cycle

Overview

Genome correspondence
Chromosome evolution
Genome rearrangements
Sorting by reversals
Genome duplication
Duplicate gene evolution
Duplication and rearrangements

Greedy Algorithms
And
Genome Rearrangements

Outline

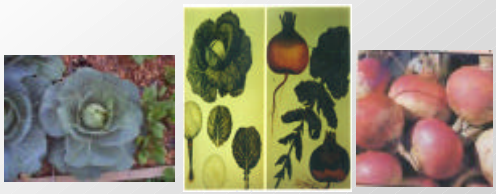
- Transforming Cabbage into Turnip
- Genome Rearrangements
- Sorting By Reversals
- Pancake Flipping Problem
- Greedy Algorithm for Sorting by Reversals
- Approximation Algorithms
- Breakpoints: a Different Face of Greed

Outline CHANGE

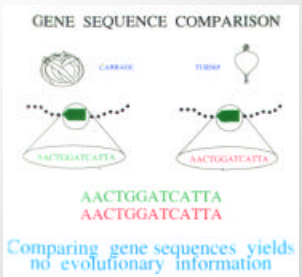
- Genome Rearrangements – give picture of splotch mouse

Turnip vs Cabbage: Look and Taste Different

- Although cabbages and turnips share a recent common ancestor, they look and taste different



Turnip vs Cabbage: Comparing Gene Sequences Yields No Evolutionary Information

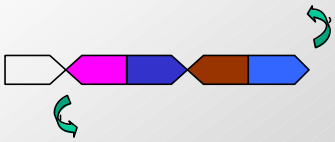


Turnip vs Cabbage: Almost Identical mtDNA gene sequences

- In 1980s Jeffrey Palmer studied evolution of plant organelles by comparing mitochondrial genomes of the cabbage and turnip
- 99% similarity between genes
- These surprisingly identical gene sequences differed in gene order
- This study helped pave the way to analyzing genome rearrangements in molecular evolution

Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:



Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:

Turnip vs Cabbage: Different mtDNA Gene Order

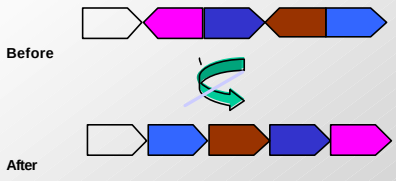
- Gene order comparison:

Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:

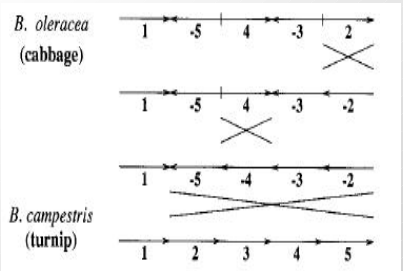
Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:

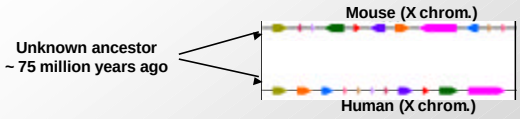


Evolution is manifested as the divergence in gene order

Transforming Cabbage into Turnip

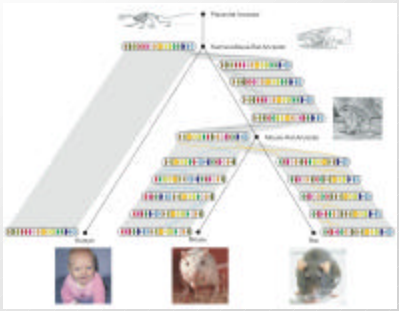


Genome rearrangements



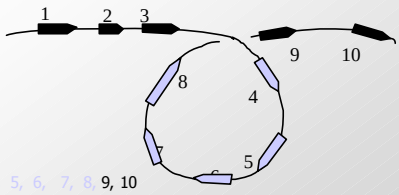
- What are the similarity blocks and how to find them?
- What is the architecture of the ancestral genome?
- What is the evolutionary scenario for transforming one genome into the other?

History of Chromosome X



Rat Consortium, *Nature*, 2004

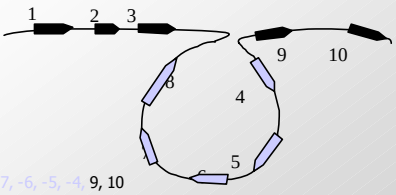
Reversals



1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.

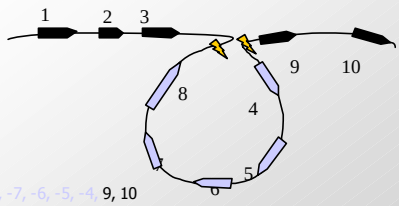
Reversals



1, 2, 3, -8, -7, -6, -5, -4, 9, 10

- Blocks represent conserved genes.
- In the course of evolution or in a clinical context, blocks 1,…,10 could be misread as 1, 2, 3, -8, -7, -6, -5, -4, 9, 10.

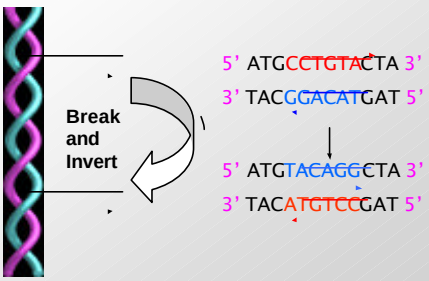
Reversals and Breakpoints



1, 2, 3, -8, -7, -6, -5, -4, 9, 10

The reversion introduced two breakpoints (disruptions in order).

Reversals: Example



Types of Rearrangements

Reversal

1 2 3 4 5 6 → 1 2 -5 -4 -3 6

Translocation

1 2 3
4 5 6 → 1 2 6
4 5 3

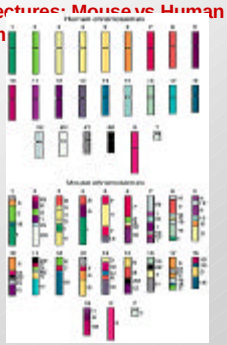
Fusion

1 2 3 4
5 6 → 1 2 3 4 5 6

Fission

Comparative Genomic Architectures: Mouse vs Human Genom

- Humans and mice have similar genomes, but their genes are ordered differently
- ~245 rearrangements
  - Reversals
  - Fusions
  - Fissions
  - Translocation



Waardenburg's Syndrome: Mouse Provides Insight into Human Genetic Disorder

- Waardenburg's syndrome is characterized by pigmentary dysphasia
- Gene implicated in the disease was linked to human chromosome 2 but it was not clear where exactly it is located on chromosome 2

Waardenburg's syndrome and splotch mice

- A breed of mice (with splotch gene) had similar symptoms caused by the same type of gene as in humans
- Scientists succeeded in identifying location of gene responsible for disorder in mice
- Finding the gene in mice gives clues to where the same gene is located in humans

Comparative Genomic Architecture of Human and Mouse Genomes

To locate where corresponding gene is in humans, we have to analyze the relative architecture of human and mouse genomes