Algorithms for Bioinformatics
Autumn 2011

VELI MÄKINEN

HTTP://WWW.CS.HELSINKI.FI/EN/COURSES/
582670/2011/S/K/1
Lecture 3

GREEDY ALGORITHMS AND GENOME REARRANGEMENTS
Background

- Genome sequencing enables us to compare genomes of two or more different species
  - \( \rightarrow \) Comparative genomics
- Basic observation:
  - Closely related species (such as human and mouse) can be almost identical in terms of genome contents...
  - ...but the order of genomic segments can be very different between species
Synteny blocks and segments

- **Synteny** – means genomic segments located on the same chromosome
  - Genes, markers (any sequence)
- **Synteny block (or syntenic block)**
  - A set of genes or markers that co-occur together in two species
- **Synteny segment (or syntenic segment)**
  - Syntenic block where the *order* of genes or markers is preserved
Synteny blocks and segments

Chromosome i, species B

Chromosome j, species C

Synteny segment

Synteny block

Homologs of the same gene
Chromosomes

- Linear chromosomes
  - Eukaryotes (mostly)

- Circular chromosomes
  - Prokaryotes (mostly)
  - Mitochondria

Also double-stranded: genes can be found on both strands (orientations)
Example: human vs mouse genome

- Human and mouse genomes share thousands of homologous genes, but they are
  - Arranged in different order
  - Located in different chromosomes

- Examples
  - Human chromosome 6 contains elements from six different mouse chromosomes
  - Analysis of X chromosome indicates that rearrangements have happened primarily *within* chromosome
Fig. 5.3. Synteny blocks shared by human and mouse X chromosomes. The arrowhead for each block indicates the direction of increasing coordinate values for the human X chromosome. Reprinted, with permission, from Pevzner P and Tesler G (2003) Genome Research 13:37–45. Copyright 2003 Cold Spring Harbor Laboratory Press.
When comparing two genomes, we can find homologous sequences in both using sequence comparison algorithms (next lecture).

This gives us a map between sequences in both genomes.
We assign numbers 1,...,n to the found homologous sequences.

By convention, we number the sequences in the first genome by their order of appearance in chromosomes.

If the homolog of i is in reverse orientation, it receives number –i (signed data).

For example, consider human vs mouse gene numbering on the right.

List order corresponds to physical order on chromosomes!
The basic data structure in the study of genome rearrangements is permutation.

A permutation of a sequence of $n$ numbers is a reordering of the sequence.

For example, 4 1 3 2 5 is a permutation of 1 2 3 4 5.
Genome rearrangement problem

- Given two genomes (set of markers), how many
  - duplications,
  - inversions and
  - translocations

do we need to do to transform the first genome to the second?

*Minimum number of operations?*
*What operations? Which order?*
Genome rearrangement problem

#duplications?
#inversions?
#translocations?

6 1 2 3 4 5

1 2 3 4 5 6
Genome rearrangement problem

Keep in mind, that the two genomes have been evolved from a common ancestor genome!
Genome rearrangements using reversals (=inversions) only

- Let’s consider a “simpler” problem where we just study **reversals** with **unsigned data**
- A reversal \( p(i, j) \) reverses the order of the segment \( \Pi_i \Pi_{i+1} \ldots \Pi_{j-1} \Pi_j \) (indexing starts from 1)
- For example, given permutation
  6 1 2 3 4 5 and reversal \( p(3, 5) \) we get permutation
  6 1 4 3 2 5

...note that we do not care about exact *positions* on the genome
Reversal distance problem

- Find the shortest **series of reversals** that, given a permutation $\Pi$, transforms it to the *identity* permutation $(1, 2, \ldots, n)$
- This quantity is denoted by $d(\Pi)$

**Reversal distance for a pair of chromosomes:**

- Find synteny blocks in both
- Number blocks in the first chromosome to identity
- Set $\Pi$ to correspond matching of second chromosome’s blocks against the first
- Find reversal distance
Solving the problem by sorting

- Our first approach to solve the reversal distance problem:
  - Examine each position $i$ of the permutation from left to right
  - At each position, if $\Pi_i \neq i$, do a reversal such that $\Pi_i = i$
- This is a *greedy* approach: we try to choose the best option at each step
Simple reversal sort: example

6 1 2 3 4 5 -> 1 6 2 3 4 5 -> 1 2 6 3 4 5 -> 1 2 3 6 4 5
-> 1 2 3 4 6 5 -> 1 2 3 4 6 5

Reversal series: p(1,2), p(2,3), p(3,4), p(4,5), p(5,6)

Is d(6 1 2 3 4 5) then 5?

6 1 2 3 4 5 -> 5 4 3 2 1 6 -> 1 2 3 4 5 6

D(6 1 2 3 4 5) = 2
How good is simple reversal sort?

- Not so good actually
- It has to do at most $n-1$ reversals with permutation of length $n$
- The algorithm can return a distance that is as large as $(n - 1)/2$ times the correct result $d(\Pi)$
  - For example, if $n = 1001$, result can be as bad as $500 \times d(\Pi)$
Computing reversals with breakpoints

- Lets investigate a better way to compute reversal distance
- First, some concepts related to permutation $\Pi_1\Pi_2, ..., \Pi_{n-1}\Pi_n$
  - Breakpoint: two elements $\Pi_i$ and $\Pi_{i+1}$ are a breakpoint, if they are not consecutive numbers
  - Adjacency: if $\Pi_i$ and $\Pi_{i+1}$ are consecutive, they are called adjacency
This permutation contains four breakpoints \textit{begin-2, 13, 58, 6-end} and five adjacencies 21, 34, 45, 87, 76
Each breakpoint in permutation needs to be removed to get to the identity permutation (=our target)

- Identity permutation does not contain any breakpoints

First and last positions special cases

- Note that each reversal can remove \textit{at most two} breakpoints

- Denote the number of breakpoints by $b(\Pi)$

\[ b(\Pi) = 4 \]

\begin{center}
\begin{tabular}{c c c c c}
2 & 1 & 3 & 4 & 5 \\
6 & 7 & 8 & 9 & 10
\end{tabular}
\end{center}
Breakpoint reversal sort

- Idea: try to remove as many breakpoints as possible (max 2) in every step

1. While $b(\pi) > 0$
2. Choose reversal $p$ that removes most breakpoints
3. Perform reversal $p$ to $\pi$
4. Output $\pi$
5. return
### Breakpoint removal: example

<table>
<thead>
<tr>
<th>Breakpoint</th>
<th>b(Π)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 2 7 6 5 1 4 3</td>
<td>6</td>
</tr>
<tr>
<td>2 8 7 6 5 1 4 3</td>
<td>5</td>
</tr>
<tr>
<td>2 3 4 1 5 6 7 8</td>
<td>3</td>
</tr>
<tr>
<td>4 3 2 1 5 6 7 8</td>
<td>2</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7 8</td>
<td>0</td>
</tr>
</tbody>
</table>
Breakpoint removal

• The previous algorithm needs refinement to be correct
• Consider the following permutation:

  1 5 6 7 2 3 4 8

• There is no reversal that decreases the number of breakpoints!
Breakpoint removal

- Reversal can always decrease breakpoint count if permutation contains *decreasing strips*

Strip: maximal segment without breakpoints

Increasing strip

Decreasing strip (including segments of length 1, except 1 and n if they are located at their correct locations)
Improved breakpoint reversal sort

1. While $b(\Pi) > 0$
2. If $\Pi$ has a decreasing strip
3. Do reversal $p$ that removes most BPs
4. Else
5. Reverse an increasing strip
6. Output $\Pi$
7. return
Is Improved BP removal enough?

- The algorithm works pretty well:
  - It produces a result that is at most four times worse than the optimal result
  - ...is this good?

- We considered only reversals
- What about translocations & duplications?
Translocations via reversals

Translocation of 2,3,4

p(2,8)
p(2,4)
p(5,8)
Genome rearrangements with reversals

- With unsigned data, the problem of finding minimum reversal distances is NP-complete.
- An algorithm has been developed that achieves 1.375-approximation (Berman et al. ESA 2002).
- However, reversal distance in signed data can be computed quickly!
  - It takes linear time w.r.t. the length of permutation (Bader, Moret, Yan, 2001).
  - We will not cover that algorithm here, but give some insight into central concepts leading to it.
We can estimate \( d(\Pi) \) by cycle decomposition.

Let's represent permutation \( \Pi = 1\ 2\ 4\ 5\ 3 \) with the following graph:

![Graph representation of permutation \( \Pi \)](image)

where edges correspond to adjacencies (identity, permutation \( F \)).
Estimating reversal distance by cycle decomposition

- **Cycle decomposition**: a set of cycles that
  - have edges with alternating colors
  - do not share edges with other cycles (=cycles are edge disjoint)
Cycle decompositions

- Let $c(\Pi)$ the maximum number of alternating, edge-disjoint cycles in the graph representation of $\Pi$
- The following formula allows estimation of $d(\Pi)$
  - $d(\Pi) \geq n + 1 - c(\Pi)$, where $n$ is the permutation length

```
0 — 1
2 — 4
5 — 3 — 6
1 — 2
4 — 5
```

$d(\Pi) \geq 5 + 1 - 4 = 2$
Cycle decompositions

- Cycle decomposition is NP-complete
- However, with signed data cycle decomposition becomes a trivial task
  - Lead also to efficient (but rather involved) reversal distance algorithms on signed data.
Cycle decomposition with signed data

- Consider the following two permutations that include orientation of markers:
  - $J$: +1  +5  -2  +3  +4
  - $K$: +1  -3  +2  +4  -5

- We modify this representation a bit to include both endpoints of each marker:
  - $J'$: 0  1a  1b  5a  5b  2b  2a  3a  3b  4a  4b  6
  - $K'$: 0  1a  1b  3b  3a  2a  2b  4a  4b  5b  5a  6
Graph representation of $J'$ and $K'$

\[ d(\Pi) \geq n + 1 - c(\Pi) = 5 + 1 - 3 = 3 \]
Reversal step 1 (ad hoc greedy algorithm)
Reversal steps 2,3,4

\[ 0 \xrightarrow{+1} 1a, \quad 1b \xrightarrow{-3} 3b, \quad 3a \xrightarrow{+2} 2a, \quad 2b \xrightarrow{-5} 5b, \quad 5a \xrightarrow{+4} 4a, \quad 4b \xrightarrow{} 6 \]

Step 2

\[ 0 \xrightarrow{+1} 1a, \quad 1b \xrightarrow{-3} 3b, \quad 3a \xrightarrow{+2} 2a, \quad 2b \xrightarrow{-4} 4b, \quad 4a \xrightarrow{-5} 5a, \quad 5b \xrightarrow{+5} 6 \]

Steps 3,4

\[ 0 \xrightarrow{+1} 1a, \quad 1b \xrightarrow{-3} 3b, \quad 3a \xrightarrow{+2} 2a, \quad 2b \xrightarrow{+4} 4a, \quad 4b \xrightarrow{-5} 5b, \quad 5a \xrightarrow{-5} 6 \]

\[ 3 \leq d(\Pi) \leq 4 \]
Multiple chromosomes

- In unichromosomal genomes, inversion (reversal) is the most common operation.
- In multichromosomal genomes, inversions, translocations, *fissions* and *fusions* are most common.
Multiple chromosomes

Let’s represent multichromosomal genome as a set of permutations, with $\$\$ denoting the boundary of a chromosome:

<table>
<thead>
<tr>
<th>Chr 1</th>
<th>Chr 2</th>
<th>Chr 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 9 $</td>
<td>1 3 2 8 $</td>
<td>7 6 4 $</td>
</tr>
</tbody>
</table>

This notation is frequently used in software used to analyse genome rearrangements.
Fusions & fissions

- Fusion: merging of two chromosomes
- Fission: chromosome is split into two chromosomes
- Both events can be represented with a translocation
Fission
Algorithms for general genomic distance problem

Human & mouse revisited

- Human and mouse are separated by about 75-83 million years of evolutionary history.
- Only a few hundred rearrangements have happened after speciation from the common ancestry.
- Pevzner & Tesler identified in 2003 for 281 synteny blocks a rearrangement from mouse to human with:
  - 149 inversions
  - 93 translocations
  - 9 fissions
Discussion

- Genome rearrangement events are very rare compared to, e.g., point mutations
  - We can study rearrangement events further back in the evolutionary history
- Rearrangements are easier to detect in comparison to many other genomic events
- We cannot detect homologs 100% correctly so the input permutation can contain errors
Two different genome rearrangement scenarios giving the same result.
GRIMM demonstration

GRIMM - Genome rearrangement algorithms

Source genome:

Destination genome:

Chromosomes:
- circular
- linear (directed)
- multichromosomal or undirected

Signs:
- signed
- unsigned

Formatting options

Report Style:
- One line per genome (chromosomes concatenated)
- One column (chromosomes separated)
- Two column before & after (chromosomes separated)
  - Horizontal
  - Vertical

Highlighting style:
- Show all possible initial steps of optimal scenarios
- Should operations (reversal, translocation, fission, fusion) be highlighted, and when?
  - before
  - after
  - between both
  - no highlighting

Chromosome end format:
- numeric (10)
- subscripts (C10)
- omit

Color coding:
- Genes should be colored according to their chromosome in which genome:
  - source
  - destination

GRIMM 1.04 by Glenn Tesler, University of California, San Diego.
Copyright © 2001-2005, The University of California.
Contains code from GRAPPA, © 2000-2001, The University of New Mexico and The University of Texas at Austin.

Glenn Tesler, GRIMM: genome rearrangements web server.
Bioinformatics, 2002
GRIMM file format

```plaintext
# useful comment about first genome
# another useful comment about it
>name of first genome
1 -4 2 $ # chromosome 1
-3 5 6 $ # chromosome 2
>name of second genome
5 -3 $
6 $
2 -4 1 $
```

GRIMM supports analysis of one, two or more genomes

http://grimm.ucsd.edu/GRIMM/grimm_instr.html
Study group assignments

MONDAY 26.9. 12-14 B222
Group 1: firstnames A - H

- Read pages 136 and 137 from Jones & Pevzner
  - Greedy approach to motif finding
- At study group, solve Problem 5.18
  - Design an input for the GreedyMotifSearch algorithm that causes the algorithm to output an incorrect result.
Group 2: firstnames I-N

- Read pages 15, 16, 19-22 (sect. 2.3) from Vazirani: Approximation algorithms, Springer 2001
  - Shortest superstring and its greedy approximations through set cover
  - (copies shared at the lecture, ask lecturer for pdf)
- At study group, present the reduction to set cover with some example
Group 3: firstnames O-Z

- Read 4 first pages of Heber & Stoye: Finding All Common Intervals of k Permutations, CPM 2001
  - An alternative way to define and compute genomic distances
  - [http://www.springerlink.com/content/ucc65djy0ft2bmq8/](http://www.springerlink.com/content/ucc65djy0ft2bmq8/)
- At study group, simulate Algorithm 1 with some example.