Lecture 3: Greedy Algorithms and Genomic Rearrangements

12.9.2013

Adapted from slides by Leena Salmela, Veli Mäkinen, Esa Pitkänen
We now have genomes of several species available

It is possible to compare genomes of two or more different species

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 – describes how genomic segments are located on the same chromosome or close to each other
  - Genes, markers (any sequence)
- Shared synteny between two species: genes are located close to each other in both of the species
- 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 block

Synteny segment

Homologs of the same gene
Chromosomes

- Linear chromosomes
  - Eukaryotes (mostly)
- Circular chromosomes
  - Prokaryotes (mostly)
  - Mitochondria
- Chromosomes are 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
Representing genomic rearrangements

- When comparing genomes, we can find homologous sequences in both using sequence comparison algorithms (next lecture).
- This gives us a map between sequences in both genomes.

Representing genomic rearrangements

- We assign numbers 1, \ldots, n to the found homologous sequences.
- By convention, we number the sequences in the first genome by their order of appearance in the 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!

<table>
<thead>
<tr>
<th>Human</th>
<th>Mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (gnat2)</td>
<td>12 (inpp1)</td>
</tr>
<tr>
<td>2 (nras)</td>
<td>13 (cd28)</td>
</tr>
<tr>
<td>3 (ngfb)</td>
<td>14 (fn1)</td>
</tr>
<tr>
<td>4 (gba)</td>
<td>15 (pax3)</td>
</tr>
<tr>
<td>5 (pklr)</td>
<td>-9 (il10)</td>
</tr>
<tr>
<td>6 (at3)</td>
<td>-8 (pdc)</td>
</tr>
<tr>
<td>7 (lamic1)</td>
<td>-7 (lamic1)</td>
</tr>
<tr>
<td>8 (pdc)</td>
<td>-6 (at3)</td>
</tr>
<tr>
<td>9 (il10)</td>
<td>...</td>
</tr>
</tbody>
</table>
Permutations

- 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 transform the first genome to the second?

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

# duplications?
# inversions?
# translocations?

5 1 2 3 4 → 1 2 3 4 5
Genome rearrangement problem

\[ \pi_1 \pi_2 \pi_3 \pi_4 \pi_5 \leftarrow \text{Permutation of } 1, \ldots, 6 \]

5 1 2 3 4 \rightarrow 1 2 3 4 5

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 $5 \ 1 \ 2 \ 3 \ 4$ and reversal $p(2, 4)$ we get permutation $5 \ 3 \ 2 \ 1 \ 4$.

Note that we do not care about the 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 reversal distance is denoted by \( d(\pi) \)

- Reversal distance for a pair of chromosomes:
  - Find synteny blocks in both
  - Number synteny blocks in the first chromosome to identity
  - Set \( \pi \) to corresponding 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 $\pi \neq i$, do a reversal such that $\pi_i = i$
- This is a *greedy* approach: we try to choose the option that looks best at the current step
Simple reversal sort: example

\[5 \ 1 \ 2 \ 3 \ 4 \quad \Rightarrow \quad 1 \ 5 \ 2 \ 3 \ 4 \quad \Rightarrow \quad 1 \ 2 \ 5 \ 3 \ 4 \quad \Rightarrow \quad 1 \ 2 \ 3 \ 5 \ 4 \quad \Rightarrow \quad 1 \ 2 \ 3 \ 4 \ 5\]

- Reversal series: \(p(1, 2), p(2, 3), p(3, 4), p(4, 5)\)
- Is \(d(5 \ 1 \ 2 \ 3 \ 4)\) then 4?
Simple reversal sort: example

5 1 2 3 4 ⇒ 1 5 2 3 4 ⇒ 1 2 5 3 4 ⇒ 1 2 3 5 4 ⇒ 1 2 3 4 5

- Reversal series: \( p(1, 2), p(2, 3), p(3, 4), p(4, 5) \)
- Is \( d(5 1 2 3 4) \) then 4?

5 1 2 3 4 ⇒ 4 3 2 1 5 ⇒ 1 2 3 4 5

- \( d(5 1 2 3 4) = 2 \)
How good is simple reversal sort?

- Not so good actually
- It has to do at most $n - 1$ reversals with permutation of length $n$
- In our previous example, the algorithm returned a distance that is as large as $(n - 1)/2$ times the correct result $d(\pi) = 2$
  - For example, if we extend the example for $n = 1001$, the result can be as bad as $500 \times d(\pi)$
Simple reversal sort is an *approximation algorithm*. It only produces an approximate solution.

- $A(\pi)$: approximate solution returned by algorithm $A$
- $OPT(\pi)$: optimal solution
- The *approximation ratio* of (minimization) algorithm $A$ is the maximum approximation ratio over *all* inputs of size $n$:

$$\max_{|\pi|=n} \frac{A(\pi)}{OPT(\pi)}$$

The approximation ratio for simple reversal sort is thus at least $(n - 1)/2$

The approximation ratio tells how much off the solution given by the algorithm can in *worst case* be from the optimal solution
Approximation ratios for maximization problems

- Previous slide gave the approximation ratio for a minimization problem like reversal distance.
- For a maximization problem (e.g. motif finding, maximizing score) the approximation ratio of an algorithm is defined as the minimum approximation ratio over all inputs of size \( n \):

\[
\min_{|\pi|=n} \frac{A(\pi)}{OPT(\pi)}
\]
Let’s investigate a better way to compute reversal distance

First some concepts related to permutation $\pi_1 \pi_2 \ldots \pi_{n-1} \pi_n$

- **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 an *adjacency*
Breakpoints and adjacencies

This permutation contains
- four breakpoints: begin-2, 13, 58, 6-end
- five adjacencies: 21, 34, 45, 87, 76
Breakpoints

- 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 \( b(\pi) \)

\[
\begin{array}{ccc}
21 & 345 & 876 \\
\end{array}
\]

\[ b(\pi) = 4 \]
Breakpoint reversal sort

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

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

\[
\begin{array}{cccccccc}
8 & 2 & 7 & 6 & 5 & 1 & 4 & 3 \\
\times
\begin{array}{cccccccc}
2 & 8 & 7 & 6 & 5 & 1 & 4 & 3 \\
\times
\begin{array}{cccccccc}
2 & 3 & 4 & 1 & 5 & 6 & 7 & 8 \\
\times
\begin{array}{cccccccc}
4 & 3 & 2 & 1 & 5 & 6 & 7 & 8 \\
\times
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
\]

\[b(\pi) = 6\]

\[b(\pi) = 5\]

\[b(\pi) = 3\]

\[b(\pi) = 2\]

\[b(\pi) = 0\]
Break point 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.

**(1 5 6 7 2 3 4 8)** → Increasing strip

**(1 5 6 7 4 3 2 8)** ← Decreasing strip (including segments of length 1, except 1 and $n$ if they are located at their correct locations)

**(1 2 3 4 7 6 5 8)**
Improved breakpoint reversal sort

1: \textbf{while} $b(\pi) > 0$ \textbf{do}
2: \hspace{1em} \textbf{if} $\pi$ has a decreasing strip \textbf{then}
3: \hspace{2em} Apply reversal $p$ such that it removes most BPs
4: \hspace{1em} \textbf{else}
5: \hspace{2em} Reverse an increasing strip
6: \hspace{1em} Output $\pi$
Is improved BP removal enough?

- The algorithm works pretty well:
  - A reversal removes at most two breakpoints
    \[ \implies \text{Optimal solution cannot be better than } b(\pi)/2 \]
  - Improved BP removal performs at most \(2 \cdot b(\pi)\) reversals
    \[ \implies \text{The result is at most } \textbf{four} \text{ times worse than the optimal} \]
    \[ \implies \text{The approximation ratio of improved BP removal is at most } 4. \]
  - Is this good?

- We considered only reversals

- What about translocations?
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)
Estimating reversal distance by cycle decomposition

- We can estimate $d(\pi)$ by cycle decomposition
- Let’s represent permutation $\pi = 1\ 2\ 4\ 5\ 3$ with the following graph

where edges correspond to adjacencies (identity, permutation $\pi$)
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)
Let $c(\pi)$ be 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

\[
\begin{align*}
0 & \rightarrow 1 \\
2 & \rightarrow 4 \\
5 & \rightarrow 3 \\
1 & \rightarrow 2 \\
4 & \rightarrow 5
\end{align*}
\]

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

- Cycle decomposition is NP-complete
- However, with signed data cycle decomposition becomes a trivial task (the cycles are vertex disjoint)
Consider the following permutation that includes orientation of the markers:
- $+1\ -5\ -3\ -2\ +4$

We modify this representation to include both endpoints of each marker:
- $0\ \text{1a}\ \text{1b}\ \text{5b}\ \text{5a}\ \text{3b}\ \text{3a}\ \text{2b}\ \text{2a}\ \text{4a}\ \text{4b}\ 6$
Graph representation of $\pi$ and identity permutation

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

$3 \leq d(\pi) \leq 4$
Reversal distance with signed data

However, the exact reversal distance in signed data can be computed quickly!

- It takes linear time w.r.t. the length of permutation (Bader, Moret, Yan 2001)
- The algorithm is quite involved
Multiple chromosomes

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

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

Human and 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 ancestor
- Pevzner and Tesler identified in 2003 for 281 syntenic 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 rearrangements 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
Outline

Biological background

Permutations and genomic rearrangements

Sorting by reversals

Simple reversal sort

Breakpoints

Cycle decomposition

Multiple chromosomes

Study group assignments
Study Group 1: (random allocation at lecture)

  - 2-approximation for sorting an unsigned permutation
  - Copies distributed at the lecture.
- In the study group
  - Go through the reasoning in the proof of Lemma 9.2.
  - Simulate the 2-approximation algorithm on the permutation

1 6 5 7 8 4 2 3 9

How many reversals does the 2-approximation algorithm need? Is this optimal?
Study Group 2: (if you did not get material at the lecture)

- 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
Study Group 3: (random allocation at lecture)

- Read pages 15, 16, 19-22 (sect. 2.3) from Vazirani: Approximation algorithms, Springer 2001
  - Shortest superstring and its greedy approximation through set cover
  - Copies distributed at the lecture.

- At the study group:
  - present the reduction to set cover with some example
  - go through the proof of Lemma 2.11