Bioinformatics 1
Biology, Sequences, Phylogenetics
Part 3
Sepp Hochreiter
Contents

4 Multiple Alignment
4.1 Motivation
4.2 Multiple Sequence Similarities and Scoring
  4.2.1 Consensus and Entropy Score
  4.2.2 Tree and Star Score
  4.2.3 Weighted Sum of Pairs Score
4.3 Multiple Alignment Algorithms
  4.3.1 Exact Methods
  4.3.2 Progressive Algorithms
  4.3.3 Other Multiple Alignment Algorithms
4.4 Profiles and Position Specific Scoring Matrices
### Motivation

Compare more than two sequences: arranged sequences so that the amino acids for every the columns match as good as possible.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>APSRKFFGCCNKKMNGRKQSLIGITNA...AKVPADTD</td>
</tr>
<tr>
<td>Chicken</td>
<td>...RKFFGCCNKKMNGDKKSLGMHTNG...AKLSADTD</td>
</tr>
<tr>
<td>Yeast</td>
<td>.GAGKTVGCCNKKCNGTLASLITEKTGVAASVDANELAKKV</td>
</tr>
<tr>
<td>E. coli</td>
<td>..ARTFVGCHNLNGSKQISKIVERNT...ASIPENV</td>
</tr>
<tr>
<td>Amoeba</td>
<td>..MRHPLMNKKLNGSRHMLVSLVSNRKELAGVAGC</td>
</tr>
<tr>
<td>Archaeon</td>
<td>AKLKEPIIAINFKTYTEATGKRAGIAKAAA...EKVYKET</td>
</tr>
</tbody>
</table>

### Example Alignment

<table>
<thead>
<tr>
<th>Human</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>VVVCAPIITA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FVPQKLD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKAQAACIQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CYKVTNG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chicken</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>VVVCAPIITA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FVPQKLD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKAQAACIQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CYKVTNG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yeast</td>
<td>50</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>VVVCAPIITA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FVPQKLD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKAQAACIQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CYKVTNG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Motivation

4 Multiple Alignment

4.1 Motivation

4.2 Scoring

4.2.1 Consensus

4.2.2 Tree and Star

4.2.3 Sum of Pairs

4.3 Algorithms

4.3.1 Exact Methods

4.3.2 Progressive

4.3.3 Other

4.4 Profiles / PSSMs
Motivation

4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

Bioinformatics 1: Biology, Sequences, Phylogenetics
Motivation

4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

Human

Chicken

Yeast

E. coli

Amoeba

Archaeon

Consensus

<table>
<thead>
<tr>
<th>Human</th>
<th>VGGASLK</th>
<th>P</th>
<th>E</th>
<th>VD</th>
<th>I</th>
<th>AKQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicken</td>
<td>VGGASLP</td>
<td>P</td>
<td>E</td>
<td>VD</td>
<td>I</td>
<td>AKH</td>
</tr>
<tr>
<td>Yeast</td>
<td>VGGASLD</td>
<td>DAAK</td>
<td>K</td>
<td>T</td>
<td>I</td>
<td>SVSEKL</td>
</tr>
<tr>
<td>E. coli</td>
<td>VGGASLE</td>
<td>P</td>
<td>E</td>
<td>VD</td>
<td>I</td>
<td>SRN</td>
</tr>
<tr>
<td>Amoeba</td>
<td>VGGASLA</td>
<td>ADA</td>
<td>A</td>
<td>V</td>
<td>VKAAEAAKQA</td>
<td></td>
</tr>
<tr>
<td>Archaeon</td>
<td>LASGVT</td>
<td>AKD</td>
<td>PEKA</td>
<td>WDLVSGI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consensus</td>
<td>vggaslK</td>
<td>.</td>
<td>ef</td>
<td>.</td>
<td>iin</td>
<td></td>
</tr>
</tbody>
</table>

Bioinformatics 1: Biology, Sequences, Phylogenetics
Motivation

Multiple sequence alignment is used to

- detect remote homologous regions
- detect motifs (regular patterns) in protein families
- detect conserved regions or positions (disulfide bonds)
- detect structural blocks like helices or sheets
- construct phylogenetic trees
- construct a profiles (search or averages)
- sequence genomes by superimposing fragments (nucleotides)
- cluster proteins according to similar regions
Scoring and Similarity

Similarity measures can be based on:

- the similarity of all sequences to a reference sequence
- the similarities between evolutionary adjacent sequences
- all pairwise similarities
4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

**consensus sequence:** obtained if for each column in the alignment the most frequent amino acid is chosen
more precisely: the amino acid or letter which has the highest score to all other amino acids or gaps in the column

**consensus score:** sum of the pairwise score between sequences and the consensus sequence

generalized by profiles instead of sequences

*profile:* relative frequency instead of most frequent
Consensus and Entropy

high entropy of the letter distribution: all letters are equally probable

zero entropy: one letter in the column

good alignment correlates with a low accumulative entropy

entropy score: \[ - \sum_i \sum_a f_{i,a} \log f_{i,a} \]

\( f_{i,a} \): relative frequency of letter \( a \) in column \( i \)
To count the number of mutations only those pairs should be compared which are evolutionary adjacent.

Evolutionary adjacent sequences are represented through a phylogenetic tree, which must be constructed.
**Tree and Star Score**

phylogenetic star: one sequence is considered as ancestor
Weighted Sum of Pairs

weighted sum of pairs: all pairwise comparisons

alignment length: \( L \)

number sequences: \( N \)

weights: reduce the influence of closely related sequences

\[
\sum_{i=1}^{L} \sum_{l=1}^{N-1} \sum_{j=l+1}^{N} w_{l,j} s(x_{i,l}, x_{i,j})
\]
Weighted Sum of Pairs

Disadvantage: relatively decreases with respect of N for conservative regions; but larger N means more conservative

\[ S_{\text{old}} = \frac{N(N-1)}{2} s(C, C) \]

\[ S_{\text{new}} = \frac{N(N-1)}{2} s(C, C) - (N-1)s(C, C) + (N-1)s(C, D) \]

\[ \frac{S_{\text{old}} - S_{\text{new}}}{S_{\text{old}}} = \frac{2(N-1)s(C, C) - 2(N-1)s(C, D)}{N(N-1)s(C, C)} = \]

\[ \frac{2}{N} \left( 1 - \frac{s(C, D)}{s(C, C)} \right) \]

for large N small difference

\[ s(C, D) < s(C, C) \]

reasonable scoring matrices:  \( 1 - \frac{s(C, D)}{s(C, C)} > 0 \)
contra-intuitive: a new letter in a column of 100 equal letters is more surprising as a new letter in a column of 3 equal letters

Information gain: \(- \log f_{i,a} = \log(N)\)

Gaps: as for pairwise algorithms, linear gaps more efficient
Multiple Alignment Algorithms

multiple alignment optimization problem: NP-hard

Exact solution: only 10 to 15 sequences

algorithm classes:

- global and progressive methods: MSA, COSA, GSA, clustalW, TCoffee

- iterative and search algorithms: DIALIGN, MultAlin, SAGA, PRRP, Realigner

- local methods (motif/profile): eMotif, Blocks, Dialign, Prosite, HMM, Gibbs sampling

- divide-and-conquer algorithms: DCA, OMA
## Multiple Alignment Algorithms

### Global progressive alignments methods

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>URL</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUSTALW</td>
<td>ftp://ftp.ebi.ac.uk/pub/software</td>
<td>Thompson et al. (1994/97)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Higgins et al. (1996)</td>
</tr>
<tr>
<td>MSA</td>
<td><a href="http://www.psc.edu/">http://www.psc.edu/</a></td>
<td>Lipman et al. (1989)</td>
</tr>
<tr>
<td></td>
<td>ftp://fastlink.nih.gov/pub/mao</td>
<td></td>
</tr>
<tr>
<td>PRALINE</td>
<td><a href="http://mathbio.nimr.mrc.ac.uk/~jhering/praline">http://mathbio.nimr.mrc.ac.uk/~jhering/praline</a></td>
<td>Heringa (1999)</td>
</tr>
</tbody>
</table>

### Iterative and search algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>URL</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIALIGN</td>
<td><a href="http://www.gsf.de/biodv/dialign.html">http://www.gsf.de/biodv/dialign.html</a></td>
<td>Morgenstern et al. (1996)</td>
</tr>
<tr>
<td></td>
<td>pub/genome/saitamacc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Projects_home_page/saga_home_page.html</td>
<td></td>
</tr>
</tbody>
</table>

### Local alignments / motif / profile

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>URL</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pub/neuwald/gibbs9_95/</td>
<td>Liu et al. (1995)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neuwald et al. (1995)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Grundy et al. (1996, 1997)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bailey and Gribskov (1998)</td>
</tr>
<tr>
<td>SAM hidden</td>
<td><a href="http://www.cse.ucsc.edu/">http://www.cse.ucsc.edu/</a></td>
<td>Krogh et al. (1994)</td>
</tr>
</tbody>
</table>
Exact Methods

MSA (Lippman et al., 1989, Gupa et al., 1995): generalizes the dynamic programming ideas from pairwise alignment

three sequences:

A-BD-E-
ACB--E-
A--DCEE
Exact Methods

memory and computational complexity: exponentially with N

Gupa et al., 1995: pairwise alignments constrain the path and not the whole hypercube must be filled

MSA (Gupa):
1. compute all pairwise alignment scores $S_{k,l}$
2. predict a phylogenetic tree based on the pairwise scores
3. compute pairwise weights based on the tree
4. construct a temporary multiple alignment with score $S_t$
5. Compute $B_{k,l}$ a lower bound on $S[k, l]$ the score of the projection of the optimal multiple alignment to $k$ and $l$
6. Compute space constraints similar to the Baum-Welch
7. compute the optimal alignment on the constraint cube; Dijkstra's shortest path algorithm for nonnegative edges; priority queue; non-negativity guarantees monotone increasing costs
8. compare the weight in the alignment with the maximal weight
last step compares actual and maximal weight, if actual is larger then a better alignment may be possible, larger maximal weight means more computational costs

Carillo-Lipman bound:

\[ B_{k,l} = S_t + S_{k,l} - \sum_{i,j} S_{i,j} \]

\[ S \geq S_t \]

\[ \sum_{i,j} S[i,j] \geq S_t \]

\[ \Rightarrow \sum_{(i,j) \neq (k,l)} S_{i,j} + S[k,l] \geq S_t \]

\[ \Rightarrow S[k,l] \geq S_t - \sum_{(i,j) \neq (k,l)} S_{i,j} \]

\[ S[k,l] \geq S_t + S_{k,l} - \sum_{i,j} S_{i,j} \]

\[ S[k,l] \geq B_{k,l} \]
Exact Methods

4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

MSA improved by the $A^*$ algorithm (Lermen and Reinert, 1997)

Algorithm 1 $A^*$-algorithm.

Input: graph (the graph), start (start node), goal (goal node), h(s) approximation of the distance of node s to the goal, S (priority queue), N (list of visited nodes)

Output: list P of the shortest path

BEGIN FUNCTION
insert (start, S)
while not isEmpty(S) do
    current_node = pop(S)
    if current_node in N then
        return “no path”
    end if
    insert (current_node, N)
    if current_node = goal then
        reconstruct_shortest_path(start, goal, graph)
    else
        {find all nodes accessible from current node}
        successors = expand(current_node, graph)
        save_predecessor_in_graph(current_node, graph)
        for all s in successors do
            predecessor(s) = current_node
            {compute and store costs}
            cost(s) = cost(current_node) + edge(graph, current_node, s)
            all_cost(s) = cost(s) + h(s)
            insert(s, S) {according to allコスト(s)}
        end for
    end if
end while
return “no path found”
END FUNCTION

BEGIN SUBFUNCTION
{shortest path P as list}
reconstruct_shortest_path (start, node, graph)
if node not= start then
    push(node, P) {get predecessor}
    predecessor = getPredecessor(node, graph)
    reconstruct_shortest_path (start, predecessor, graph)
else
    return P
end if
END SUBFUNCTION
Exact Methods

4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

MSA: weighted sum of pairs and a linear gap penalty
Weight: difference pairwise and projected multiple alignment (larger difference means higher weight)

similar sequences: pull the multiple alignment towards them which down-weights them

weights through the phylogenetic tree remove weights between distant sequences

Summing up all the weights: overall divergence of the sequences
Progressive methods are the most popular methods for multiple alignment: ClustalW (Thomson, Higgins, Gibson, 1994) and TCoffee (Notredame, Higgins, Heringa, 2000)

ClustalW and TCoffee:
- perform pairwise alignment for each pair
- weight matrix: one minus the ratio of perfect matches
- construct a phylogenetic tree (Neighbor-Joining method)
- alignments between pairs sequences/alignments (start with closest distance); alignments are propagated through the tree

Initial alignments may be found through local alignment

phylogenetic tree supplies the weighting factors
Disadvantage progressive methods:

- local minima
- same scoring matrix for close and remote related sequences and same gap parameters

**ClustalW**

gap penalties context dependent:

- gaps in hydrophobic regions are more penalized
- gaps which are within eight amino acids to other gaps are more penalized
- gaps in regions of other gaps have lower gap opening penalty
- gap penalties are amino acid dependent
scoring matrices are adapted:
  ➡ scoring matrix from the PAM or the BLOSUM families

sequences are weighted through a phylogenetic tree:
  ➡ similar sequences lower weights (unbalanced data sets)
  ➡ phylogenetic tree weights with $w_i$ as the weight of sequence $i$

$$
\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w_i w_j s(i, j)
$$

adaptive phylogenetic tree:
  ➡ insufficient scores change the tree

initial gap penalty parameters:
  ➡ according to scoring matrix
  ➡ similarity of the sequences (% identity)
  ➡ length of the sequences (log of the shorter sequences is added)
  ➡ difference of the length to avoid gaps in the shorter sequence

$$
(1 + |\log(n/m)|)
$$
Progressive Methods

TCoffee (Tree based Consistency Objective Function For alignment Evaluation) often better alignment than clustalW

TCoffee work as follows:

- libraries of pairwise alignments based on both global (clustalW) and local (FASTA) alignments (combination is more reliable)

- library weights are computed according to % identity

- libraries are combined and extended; arithmetic mean of weights; extension by aligning two sequences through a third sequence

- progressive alignment with a distance based on extended library
Center Star Alignment

center sequence \( \bar{i} \):  
\[
\bar{i} = \arg \min_i \sum_j C(i, j)
\]

pairwise alignment costs \( C(i, j) \)

\[
\bar{i} = 1
\]

new sequence is added to the set of aligned sequences by a pairwise alignment to the center sequence introducing new gaps
Other Methods

Gusfield, 1993: cost is less then twice as of the optimal cost, if

\[ C'(i, i) = 0 \quad \text{and} \quad C'(i, j) \leq C'(i, k) + C'(k, j) \]

scoring matrix \( s \) with

\[
\begin{align*}
& s(\cdot, \cdot) = 0 \\
& s(\cdot, i) < 0 \quad \text{with} \quad A \quad B \\
& s(k, k) \geq s(i, k) + s(k, j) - s(i, j) \quad \text{for each} \quad A \quad C \quad C \quad A
\end{align*}
\]

Then \( C'(i, j) = S_{i,i} - 2 S_{i,j} + S_{j,j} \) fulfills above conditions

The second conditions is

\[
\begin{align*}
S_{i,i} - 2 S_{i,j} + S_{j,j} & \leq S_{i,i} - 2 S_{i,k} + S_{k,k} + S_{k,k} - 2 S_{k,j} + S_{j,j} \\
\iff S_{i,j} & \geq S_{i,k} + S_{k,j} - S_{k,k}
\end{align*}
\]
align i to k and j to k then align i, j, and k based on the pairwise alignments, the alignment has a gap if a gap was in one alignment

S is score of the multiple alignment

Per construction: $S[i, k] = S_{i,k}, S[k, j] = S_{k,j}$ and $S[k, k] = S_{k,k}$

Componentwise holds: $s(i, j) \geq s(i, k) + s(k, j) - s(k, k)$

Therefore $S[i, j] \geq S[i, k] + S[k, j] - S[k, k]$ and $S[i, j] \geq S_{i,k} + S_{k,j} - S_{k,k}$

inequality to show follows from $S_{i,j} \geq S[i, j]$
idea of the proof of Gusfield center sequence alignment with cost $C$ and the optimal cost $C^*$

$$C = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} C(i, j) \leq$$

$$\sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} C(i, 1) + C(1, j) = 2(N - 1) \sum_{i=2}^{N} C(i, 1)$$

$$C^* = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} C(i, j) \geq$$

$$\sum_{i=1}^{N} \sum_{j=2}^{N} C(i, 1) = N \sum_{i=2}^{N} C(i, 1)$$

$$\Rightarrow \frac{C}{C^*} \leq \frac{2(N - 1)}{N} \leq 2$$
Motifs or pattern can be superimposed for alignment landmarks

Profiles and blocks can be derived from multiple alignments
Other Methods

4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

SAGA (Sequence Alignment by Genetic Algorithm): genetic algorithm

MSASA (Multiple Sequence Alignment by Simulated Annealing): simulated annealing

Gibbs sampling

HMMs (hidden Markov models) can be used to find motifs
Other Methods

4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs
Profiles and PSSMs

Profiles and Position Specific Scoring Matrices

Assumptions:
- \( \mathbf{x} \) is i.i.d. in its elements according to \( p_x \)
- \( n \) the length of \( \mathbf{x} \) is large
- expected letter score for random sequences \( \sum_i p_x(i) s(i) < 0 \)
- exist \( i \) for which \( s(i) > 0 \)

\[
S_n = \sum_{i=1}^{n} s(i) \quad \text{centered value: } \tilde{S}_n = S_n - \frac{\ln n}{\lambda}
\]

\[
P\left(\tilde{S}_n > S\right) \approx 1 - \exp\left(-K e^{-\lambda S}\right) \approx K e^{-\lambda S}
\]

\[
\sum_i p_x(i) \exp(\lambda s(i)) = 1
\]
4 Multiple Alignment
4.1 Motivation
4.2 Scoring
4.2.1 Consensus
4.2.2 Tree and Star
4.2.3 Sum of Pairs
4.3 Algorithms
4.3.1 Exact Methods
4.3.2 Progressive
4.3.3 Other
4.4 Profiles / PSSMs

$q_i$: frequency of a letter $a_i$ in a column of a multiple alignment

for sufficient high scoring segments

$$\lim_{n \to \infty} q_i = p_x(i) \exp(\lambda \: s(i))$$

$$\Rightarrow s(i) = \ln \left( \frac{q_i}{p_x(i)} \right) / \lambda$$

“Position Specific Scoring Matrices” (PSSMs) or profiles

new sequence: high scores mean similar alignment sequences