Announcements

• Reminder HW #3 due on Thursday (*)
Starting point: DP is too slow!

- Given $k$ sequences of length $n$:
  - Space for matrix: $O(n^k)$
  - Neighbors/cell: $2^k - 1$
  - Time to compute SP score: $O(k^2)$
  - Overall runtime: $O(k^2 2^k n^k)$

- Can align about 7 relatively short (length 200 - 300) sequences in a reasonable amount of time

- $2^7 \times 200^7 > 1,600,000,000,000,000,000,000$
  - Exact optimality is generally not attainable
Heuristics for multiple sequence alignment

• Exact optimality is too slow, even by DP
  • Alternative: Seek good suboptimal solutions that are attainable in reasonable time

• Key questions:
  • What is a “good” suboptimal solution?
  • What is “reasonable” time?

• Heuristics focus on reduction of search space
  • Divide-and-conquer alignment
  • Greedy alignment (progressive)
Progressive alignment

• Idea:
  • Build MSA from a series of pairwise alignments

• Strategy:
  • Choose two sequences to align (optimally)
  • Hold PSA fixed, treat as a new sequence, and iterate

• For k sequences, require k - 1 PSAs

• Does the order matter?
  • What criteria are used to choose the sequences?
Complexity of progressive alignment

• Given $k$ sequences of length $n$, we need:
  • $O(k^2n^2)$ to compute the alignments
  • $O(kn^2)$ to build multiple alignment
  • $O(\ ?)$ to build the guide tree (depends on approach)

• Overall: $O(k^2n^2)$
  • Compare with dynamic programming: $O(k^22^kn^k)$

• BUT
  • Is there any guarantee on the quality of the progressive MSA?
Shortcomings of progressive approaches

- Progressive MSA strongly dependent upon initial alignments

- If sequences aligned at each step are similar
  - Progressive approach works well

- If MSA is built on dubious PSAs
  - Errors in alignment propagated and amplified

- Post-processing solution:
  - Iterative refinement
Dangers of progressive alignment

- Initial alignments are “frozen” even when new evidence is introduced

- Example:

  - $x$: GAAGTT
  - $y$: GAC–TT
  - $z$: GAACCTG
  - $w$: GTACTG

  Frozen by initial PSA

  Additional sequences make clear that $y$: GA–CTT
Iterative refinement of progressive MSA

- For each \( j = 1 \) to \( N \)
  - Remove sequence \( x^j \) and realign to remaining alignment of \( x^1, \ldots, x^{j-1}, x^{j+1}, \ldots, x^N \)

- Repeat until alignment converges
Ex: Iterative refinement

• Progressive alignment \((x, y), (z, w), (xy, zw)\):

  \[
  \begin{align*}
  x & : \quad \text{GAAGTTA} \\
  y & : \quad \text{GAC}-\text{TTA} \\
  z & : \quad \text{GAACGTGA} \\
  w & : \quad \text{GTACTGA}
  \end{align*}
  \]

• After realigning \(y\) to the remainder:

  \[
  \begin{align*}
  x & : \quad \text{GAAGTTA} \\
  y & : \quad \text{G–ACTTA} \\
  z & : \quad \text{GAACGTGA} \\
  w & : \quad \text{GTACTGA} \\
  \end{align*}
  \]

  + 3 matches
Limitations of iterative refinement

• Example not handled well:

\[
\begin{align*}
x & : \text{ GAAGTTA } \\
y_1 & : \text{ GAC–TTA } \\
y_2 & : \text{ GAC–TTA } \\
y_3 & : \text{ GAC–TTA } \\
z & : \text{ GAACCTGA } \\
w & : \text{ GTACTGA }
\end{align*}
\]

Realigning any single \( y_i \) changes nothing
Progressive alignment and consistency

• Progressive alignment fails when errors are fixed in early pairwise alignments

• Post-processing (e.g. iterative refinement) attempts to remedy this after the fact

• Alternatively, pre-processing can prevent progressive alignment errors before they occur

• One popular approach is consistency refinement
Consistency

- Consistency idea:
  - MSA should strive to be consistent with pairwise sequence alignments

- Rather than refine the MSA after the fact
  - Consistency operates on initial pairwise alignments
  - Uses all sequences to decide how each sequence pair should be aligned
Consistency

• Rather than directly align $x$ to $y$
  • Consider alignments through remaining sequences $z$

• Idea: Align $x$ to $z$ and $z$ to $y$, get an alignment of $x$ to $y$
Consistency

- Alignment of $x$ to $y$ through $z$ reveals better alignment of $x$ to $y$
Consistency

- Potential error avoided in initial pairwise alignment
- Results in a better progressive multiple sequence alignment
Ex: MSA inconsistent with optimal PSAs

- Error made in initial alignment between $x$ and $y$
- Propagated in MSA obtained by progressive alignment

\[
\begin{align*}
  x &: \text{GAAGTTA} \\
  y &: \text{GAC-TTA} \\
  z &: \text{GAACTGA} \\
  w &: \text{GTACTGA}
\end{align*}
\]
Ex: Error avoided by consistency

• Initially, C in y can align to either A or G in x

\[
y: \text{GA}C\text{TTA} \\
x: \text{GAAGTTA}
\]

• To avoid initial error, rather than simply aligning x to y, consider aligning x to y through z

\[
y: \text{GA}C\text{TTA} \\
z: \text{GAACCTGA} \\
x: \text{GAAGTTA}
\]  
\[
y: \text{GA}C\text{TTA} \\
x: \text{GAAGTTA} \\
z: \text{GAACCTGA} \\
w: \text{GTACCTGA}
\]
COFFEE maximizes consistency with a library

**COFFEE: an objective function for multiple sequence alignments**

Cédric Notredame¹, Liisa Holm¹ and Desmond G. Higgins²

**Motivation:** In order to increase the accuracy of multiple sequence alignments, we designed a new strategy for optimizing multiple sequence alignments by genetic algorithm. We named it COFFEE (Consistency based Objective Function For alignmEnt Evaluation). The COFFEE score reflects the level of consistency between a multiple sequence alignment and a library containing pairwise alignments of the same sequences.

- COFFEE is a consistency-based **objective function**
- Objective function is target for optimization
Libraries of initial pairwise alignments

- Strategies to obtain PSAs for progressive MSA:
  - ClustalW uses variant Smith-Waterman local alignment
  - PileUp uses Needleman-Wunsch global alignment

- T-Coffee Idea:
  - Use a mixture of local and global alignments

- Goal:
  - Retain advantages of both methods
Integrating several methods with consistency

Figure 4. Layout of T-Coffee.

This figure indicates the layout of T-Coffee. Local and global pairwise alignments are first computed and then combined into a primary library that is extended in order to be used for computing the multiple sequence alignment in a progressive manner.

T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment

Cédric Notredame¹,²,³*, Desmond G. Higgins⁴ and Jaap Heringa¹
T-Coffee

- Uses both local and global alignment approaches
  - Consistency checked against both libraries of PSAs

- Combines different multiple alignment methods
  - Idea: Methods suggest consensus multiple alignment

- Combines sequence alignment methods with structural alignment techniques

- Allows input of user knowledge
Schematic of T-Coffee

• Two primary libraries are constructed
  • One global and one local

This figure indicates the layout of T-Coffee. Local and global pairwise alignments are first computed and then combined into a primary library that is extended in order to be used for computing the multiple sequence alignment in a progressive manner.
Construction of primary libraries

• Libraries contain information on all \( \binom{k}{2} \) PSAs

Global alignments

<table>
<thead>
<tr>
<th>A</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local alignments

<table>
<thead>
<tr>
<th>A</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• ClustalW

• LAlign
• Top-10 non-overlapping
Extension: Consistency refinement of PSAs

- Consider alignment of A to B through remainder:
  - A to B, A to B through C, A to B through D

- Alignments are combined and then resolved by DP
  - Resulting extended library used in progressive alignment
Check please: Running time of T-Coffee

- Given $k$ sequences of length $n$:
  - Pairwise library computation: $O(k^2n^2)$
  - Library extension: $O(k^3n)$
  - Computation of guide tree: $O(k^3)$
  - Progressive alignment: $O(kn^2)$
- Overall: $O(k^2n^2) + O(k^3n) + O(k^3) + O(kn^2)$
Comparison of T-Coffee and ClustalW

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Test 5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClustalW</td>
<td>78.5</td>
<td>32.2</td>
<td>42.5</td>
<td>65.7</td>
<td>74.3</td>
<td>66.4</td>
</tr>
<tr>
<td>T-Coffee</td>
<td>80.7</td>
<td>37.3</td>
<td>52.9</td>
<td>83.2</td>
<td>88.7</td>
<td>72.1</td>
</tr>
</tbody>
</table>

- In-depth comparison of MSA performance:
  - http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/

A comprehensive comparison of multiple sequence alignment programs

Julie D. Thompson, Frédéric Plewniak and Olivier Poch*

© 1999 Oxford University Press
## Performance of different alignment tools

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BALiBASE (237)</th>
<th></th>
<th>PREFAB (1932)</th>
<th></th>
<th>SABmark (698)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q</td>
<td>t</td>
<td>Q</td>
<td>t</td>
<td>Q</td>
<td>t</td>
</tr>
<tr>
<td>Align-m</td>
<td>0.804</td>
<td>19:25</td>
<td>-</td>
<td>-</td>
<td>0.352</td>
<td>56:44</td>
</tr>
<tr>
<td>DIALIGN</td>
<td>0.832</td>
<td>2:53</td>
<td>0.572</td>
<td>12:25:00</td>
<td>0.410</td>
<td>8:28</td>
</tr>
<tr>
<td>CLUSTALW</td>
<td>0.861</td>
<td>1:07</td>
<td>0.589</td>
<td>2:57:00</td>
<td>0.439</td>
<td>2:16</td>
</tr>
<tr>
<td>MAFFT</td>
<td>0.882</td>
<td>1:18</td>
<td>0.648</td>
<td>2:36:00</td>
<td>0.442</td>
<td>7:33</td>
</tr>
<tr>
<td>T-Coffee</td>
<td>0.883</td>
<td>21:31</td>
<td>0.636</td>
<td>144:51:00</td>
<td>0.456</td>
<td>59:10</td>
</tr>
<tr>
<td>MUSCLE</td>
<td>0.896</td>
<td>1:05</td>
<td>0.648</td>
<td>3:11:00</td>
<td>0.464</td>
<td>20:42</td>
</tr>
<tr>
<td>PROBCONS</td>
<td>0.910</td>
<td>5:32</td>
<td>0.668</td>
<td>19:41:00</td>
<td>0.505</td>
<td>17:20</td>
</tr>
</tbody>
</table>
ProbCons

- ProbCons appears to give the most accurate MSAs
  - Probabilistic Consistency-based MSA
  - Uses HMM to predict the probability of an alignment

Resource

ProbCons: Probabilistic consistency-based multiple sequence alignment

Chuong B. Do,¹ Mahathi S.P. Mahabhashyam,¹ Michael Brudno,¹ and Serafim Batzoglou¹.²

¹Department of Computer Science, Stanford University, Stanford, California 94305, USA

To study gene evolution across a wide range of organisms, biologists need accurate tools for multiple sequence alignment of protein families. Obtaining accurate alignments, however, is a difficult computational problem because of not only the high computational cost but also the lack of proper objective functions for measuring alignment quality. In this paper, we introduce probabilistic consistency, a novel scoring function for multiple sequence comparisons. We present ProbCons, a practical tool for progressive protein multiple sequence alignment based on probabilistic consistency, and evaluate its performance on several standard alignment benchmark data sets. On the BAliBASE, SABmark, and PREFAB benchmark alignment databases, ProbCons achieves statistically significant improvement over other leading methods while maintaining practical speed. ProbCons is publicly available as a Web resource.
Recall: HMM formulation of PSA

- Three states corresponding to three possibilities:
  - **M**: Two residues align
  - **Y**: Residue in sequence 2 aligns to gap in sequence 1
  - **X**: Residue in sequence 1 aligns to gap in sequence 2

- Think generative:
  - Each of these possibilities should correspond to a state
  - State **M**, **Y**, **X** emit aligned pairs
Use of HMM for PSA by ProbCons

- HMM specifies probability distribution over all alignments between a pair of sequences $x$ and $y$
  - Emissions based on BLOSUM62
  - Transitions estimated by EM

- Because HMM places probability over alignments
  - Can compute probability that residues $x_i$ and $y_j$ are paired in an alignment generated by the model
ProbCons maximizes expected accuracy

• Expected accuracy of PSA between \( x \) and \( y \)
  • Average frequency that paired residue in PSA is correct
  • Why maximize expected accuracy instead of likelihood?

\textit{Step 2: Computation of expected accuracies}

Define the expected accuracy of a pairwise alignment \( a \) between \( x \) and \( y \) to be the expected number of correctly aligned pairs of letters, divided by the length of the shorter sequence:

\[
E_{a^*}(\text{accuracy}(a,a^*)|x,y) = \frac{1}{\min\{|x|,|y|\}} \sum_{x_i \sim y_j \in a} P(x_i \sim y_j \in a^*|x,y).
\]

For each pair of sequences \( x, y \in S \), compute the alignment \( a \) that maximizes expected accuracy by dynamic programming, and set \( E(x, y) = E_{a^*}(\text{accuracy}(a, a^*)|x, y) \).
Discussion: Choosing a good PSA

• Recall purpose of Viterbi algorithm
  • For set of emissions, find most like parse through HMM

• For sequences $x$ and $y$
  • Viterbi picks alignment that is most likely to be optimal

• However:
  • The most *likely* alignment may not be the most *accurate*!

• Alternative:
  • Find the alignment with maximum expected accuracy
Lazy teacher analogy

• 10 students take a 10-question true-false quiz
  • How do you make the answer key?

• Viterbi says to use answer sheet of the best student
  • MEA says to take a weighted majority vote
Choice of PSA: Viterbi versus MEA

• Viterbi
  • Picks the alignment with the highest chance of being completely correct

• Maximum Expected Accuracy
  • Picks the alignment with the highest expected number of correct predictions

• Which one produces the higher quality alignment?
Evaluating methods of MSA

• Since DP for MSA is prohibitively slow
  • Heuristic methods are required

• Heuristics vary in both speed and accuracy

• Speed well quantified by computational complexity
  • How can accuracy be quantified?

• Idea: Construct benchmark sets of “correct” MSAs
  • Evaluate ability to reconstruct “correct” alignment
  • Metric: $Q = \text{proportion of correctly aligned residues}$
## Performance of different alignment tools

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BALiBASE (237)</th>
<th></th>
<th>PREFAB (1932)</th>
<th></th>
<th>SABmark (698)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q</td>
<td>t</td>
<td>Q</td>
<td>t</td>
<td>Q</td>
<td>t</td>
</tr>
<tr>
<td>Align-m</td>
<td>0.804</td>
<td>19:25</td>
<td>-</td>
<td>-</td>
<td>0.352</td>
<td>56:44</td>
</tr>
<tr>
<td>DIALIGN</td>
<td>0.832</td>
<td>2:53</td>
<td>0.572</td>
<td>12:25:00</td>
<td>0.410</td>
<td>8:28</td>
</tr>
<tr>
<td>CLUSTALW</td>
<td>0.861</td>
<td>1:07</td>
<td>0.589</td>
<td>2:57:00</td>
<td>0.439</td>
<td>2:16</td>
</tr>
<tr>
<td>MAFFT</td>
<td>0.882</td>
<td>1:18</td>
<td>0.648</td>
<td>2:36:00</td>
<td>0.442</td>
<td>7:33</td>
</tr>
<tr>
<td>T-Coffee</td>
<td>0.883</td>
<td>21:31</td>
<td>0.636</td>
<td>144:51:00</td>
<td>0.456</td>
<td>59:10</td>
</tr>
<tr>
<td>MUSCLE</td>
<td>0.896</td>
<td>1:05</td>
<td>0.648</td>
<td>3:11:00</td>
<td>0.464</td>
<td>20:42</td>
</tr>
<tr>
<td>PROBCONS</td>
<td>0.910</td>
<td>5:32</td>
<td>0.668</td>
<td>19:41:00</td>
<td>0.505</td>
<td>17:20</td>
</tr>
</tbody>
</table>
Probabilistic consistency transform

• Consider alignment of $x$ to $y$ through remainder:

**Step 3: Probabilistic consistency transformation**

Reestimate the match quality scores $P(x_i \sim y_j \in a^* | x, y)$ by applying the *probabilistic consistency transformation*, which incorporates similarity of $x$ and $y$ to other sequences from $S$ into the $x$–$y$ pairwise comparison:

$$P'(x_i \sim y_j \in a^* | x, y) \leftarrow \frac{1}{|S|} \sum_{z \in S} \sum_{z_k} P(x_i \sim z_k \in a^* | x, z) P(z_k \sim y_j \in a^* | z, y).$$

In matrix form, the transformation may be written as

$$P'_{xy} \leftarrow \frac{1}{|S|} \sum_{z \in S} P_{xz} P_{zy}.$$
Next: Create of guide tree for progressive MSA

• Consistency yields extended library of PSAs
  • Next create guide tree using expected accuracy as measure of similarity

• Finally, use guide tree to create progressive MSA
  • Post-processing option using iterative refinement
Tree-based iterative refinement

- Randomly partition alignment into two groups
  - Sever branch on guide tree

- Fix alignments within each group
  - Realign groups to one another
Summary: ProbCons

- ProbCons is designed to maximize accuracy
  - Empirically validated against multiple alignment datasets

- Running time is not too bad: $O(n^3)$

- Useful point:
  - No use of protein-specific alignment information

- ProbCons can be used to align DNA sequences
  - Improved accuracy over Needleman-Wunsch
Consistency-based progressive methods

- T-Coffee
  - Rather than using one initial set optimal PSAs, uses combination of global and local alignments
  - Refines set of initial alignments using consistency
  - Running time: $O(k^2n^2) + O(k^3n)$

- ProbCons
  - Uses probabilistic consistency with pair-HMM
  - Maximizes expected accuracy rather than likelihood
  - Running time: $O(n^3)$
MUSCLE

• MUSCLE is fast and scalable to large problems

DOI: 10.1093/nar/gkh340

MUSCLE: multiple sequence alignment with high accuracy and high throughput

Robert C. Edgar*

195 Roque Moraes Drive, Mill Valley, CA 94941, USA
MUSCLE

- MUSCLE uses two-pass progressive alignment strategy

- Three stages
  - Draft progressive alignment
  - Improved progressive alignment
  - Iterative refinement

- Algorithm may be terminated at end of each stage
  - Each stage yield a multiple sequence alignment
  - Subsequent stages aim to improve, cost running time
The problem of scale

• For some applications, $O(n^3)$ is certainly too slow
  • How can longer sequences be more rapidly aligned?

• Progressive approach first finds all optimal PSAs
  • Initial step already $O(k^2n^2)$, can this be improved upon?

• Idea: How do BLAST, FASTA accelerate local PSA?

• Rather than initially forming optimal PSAs
  • MUSCLE finds short exact matches between pairs
  • Speeds up MSA, allowing for alignment on larger scale
Stage 1 MSA: Fastest, least accurate

Stage 2 MSA: Slower, more accurate

Stage 3 MSA: Slowest, most accurate
Analysis of MUSCLE algorithm

• For alignment of $k$ sequences of length $n$

• Space complexity: $O(k^2+n^2)$

• Time complexity: $O(k^4+kn^2)$
  • Time complexity without refinement (omit stage 3): $O(k^3+kn^2)$

• Compare to previous methods w/ min $O(k^2n^2)$
  • Fast enough to align 5000 sequences of length 350 in about 7 minutes on a desktop computer
Recap of methods for MSA

• **ClustalW:** Most widely used
  - [http://www.ebi.ac.uk/clustalw/](http://www.ebi.ac.uk/clustalw/)

• **T-Coffee:** Better but slower
  - [http://www.ch.embnet.org/software/TCoffee.html](http://www.ch.embnet.org/software/TCoffee.html)

• **ProbCons:** Most accurate

• **MUSCLE:** Most scalable
  - Two-pass progressive method
  - [http://phylogenomics.berkeley.edu/cgi-bin/muscle/input_muscle.py](http://phylogenomics.berkeley.edu/cgi-bin/muscle/input_muscle.py)
## Performance of different alignment tools

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>BALiBASE (237)</th>
<th></th>
<th>PREFAB (1932)</th>
<th></th>
<th>SABmark (698)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q t</td>
<td></td>
<td>Q t</td>
<td></td>
<td>Q t</td>
<td></td>
</tr>
<tr>
<td>Align-m</td>
<td>0.804 19:25</td>
<td>-</td>
<td>-</td>
<td></td>
<td>0.352 56:44</td>
<td></td>
</tr>
<tr>
<td>DI ALIGN</td>
<td>0.832 2:53</td>
<td>0.572 12:25:00</td>
<td>0.410 8:28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLUSTALW</td>
<td>0.861 1:07</td>
<td>0.589 2:57:00</td>
<td>0.439 2:16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAFFT</td>
<td>0.882 1:18</td>
<td>0.648 2:36:00</td>
<td>0.442 7:33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-Coffee</td>
<td>0.883 21:31</td>
<td>0.636 144:51:00</td>
<td>0.456 59:10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUSCLE</td>
<td>0.896 1:05</td>
<td>0.648 3:11:00</td>
<td>0.464 20:42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROBCons</td>
<td>0.910 5:32</td>
<td>0.668 19:41:00</td>
<td>0.505 17:20</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>