5 Multiple Match Refinement and T-Coffee

In this lecture we desribe a popular multiple sequence program, T-Coffee.

This exposition is based on the following sources, which are all recommended reading:

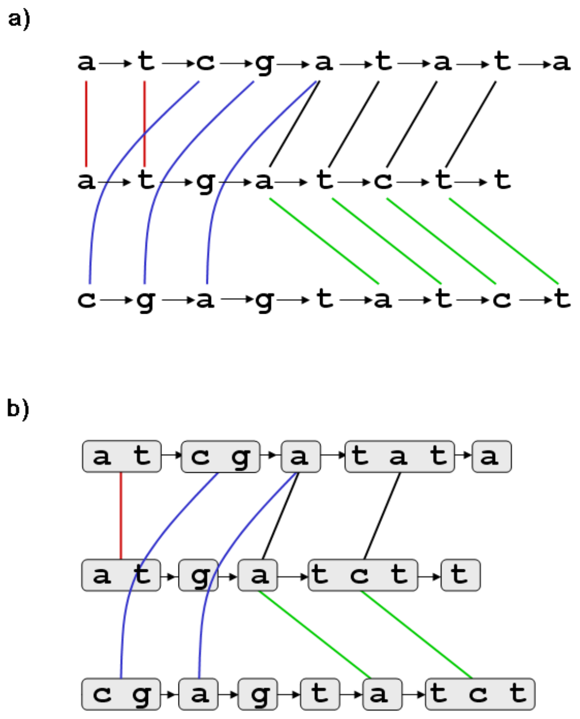
1. Notredame, Higgins, Heringa: T-Coffee, a Novel Method for Fast and Accurate Multiple Sequence Alignment, Journal of Molecular Biology, 2000, Vol 302, pages 205-217.
2. Rausch, Emde, Weese, Döring, Notredame, Reinert: Segment-based multiple sequence alignment, ECCB 2008, Cagliari

5 Multiple Match refinement

We will not deal in detail with algorithms for match refinement, but introduce the concept informally using the following example.

The result of the multiple match refinement naturally induces an input graph for the *multiple trace problem* which is NP-hard but could be solved with ILP based techniques.

However, in practice we can resort to more efficient, but heuristic methods. In the second part of the lecture we will describe the TCoffee algorithm which is basically a heuristic for the multiple trace problem (although originally not advertised as such).



5.1 T-Coffee

We now discuss a simple multiple alignment algorithm that outperforms ClustalW on certain data sets, namely T-Coffee. T-Coffee stands for **T**ree based **C**onsistency **O**bjective **F**unction **F**or alignm**E**nt **E**valuation. Its basic idea consists of combining global and local sequence information.

It uses the same progressive alignment method as ClustalW but tries to rectify the greedy choices made by incorporating information from *all* pairs of sequences.

The basic steps of T-Coffee are:

1. Generate *primary* libraries of alignments.
2. Derive library *weights*.
3. *Combine* libraries into single primary library.
4. *Extend* the library.
5. Use the extended library for progressive alignment.

5.2 Generating primary libraries

A primary library in T-Coffee contains a set of pairwise alignments and could be derived using any method. By default two libraries are generated:

1. A library of all global pairwise alignments using ClustalW.
2. A library of the ten top-scoring local alignments using Lalign from the FASTA package.

All these alignments contain information that is more or less reliable. Hence T-Coffee combines them to produce more reliable alignments.

5.3 Derive library weights

Each library is weighted with the percent identity, a measure which is known to be a reasonable indicator when aligning sequences with more than 30% identity.

We give now an example for computing a global primary library and weighting it. Assume we are given the four strings:

```
s1 = GARFIELD THE LAST FAT CAT
s2 = GARFIELD THE FAST CAT
s3 = GARFIELD THE VERY FAST CAT
s4 = THE FAT CAT
```

The regular progressive alignment method would produce the following alignment:

```
    a1 = GARFIELD THE LAST FA-T CAT
    a2 = GARFIELD THE FAST CA-T
    a3 = GARFIELD THE VERY FAST CAT
    a4 = -------- THE ---- FA-T CAT
```

Obviously the second `CAT` is not correctly aligned. T-Coffee would first produce the following global, primary library.

The percent identity is computed on *matching* positions. (Ignore the blanks, they are inserted for better readibility.)

```
a1=GARFIELD THE LAST FAT CAT        a2=GARFIELD THE ---- FAST CAT
a2=GARFIELD THE FAST CAT ---        a3=GARFIELD THE VERY FAST CAT
     weight=88                           weight=100

a1=GARFIELD THE LAST FA-T CAT       a2=GARFIELD THE FAST CAT
a3=GARFIELD THE VERY FAST CAT       a4=-------- THE FA-T CAT
     weight=80                           weight=100

a1=GARFIELD THE LAST FAT CAT        a3=GARFIELD THE VERY FAST CAT
a4=-------- THE ---- FAT CAT        a4=-------- THE ---- FA-T CAT
     weight=100                          weight=100
```

5.4 Combine libraries

If there is more than one primary library available, they are combined by merging any pair that is duplicated between the two libraries. The combined pair has as a new weight the sum of the two individual weights.

After combining the primary libraries into a single primary library, this library is extended by incorporating consistency information.

5.5 Extending the primary library

Finding the highest weighted, consistent subsets of pairwise contraints is known as the *maximum weight trace* problem and is by itself NP-hard.

Hence T-Coffee employs a heuristic strategy with the goal to compute weights that reflect the information contained in the whole library. To do so it employs a *triplet* approach. This works as follows for each pair of sequences s_i, s_j.

1. Align s_i and s_j *through* another sequence s_l, $1 \leq l \neq i, j \leq k$. For each column c of the alignment $A(s_i, s_j)$ that contains no gap character, we set the initial weight of the match $A[i,c] \leftrightarrow A[j,c]$ to the primary library's weight of the alignment of the x-th character of s_i with the y-th character of s_j. The weight of columns containing gap characters is set to 0.
2. Take the minimum weight of the primary library alignments between s_i, s_l and s_l, s_j if the x-th character of s_i is aligned to the z-th character of s_l and the z-th character of s_l is aligned to the y-th of s_j. The weight is added to the inital weight of the match $x \leftrightarrow y$.

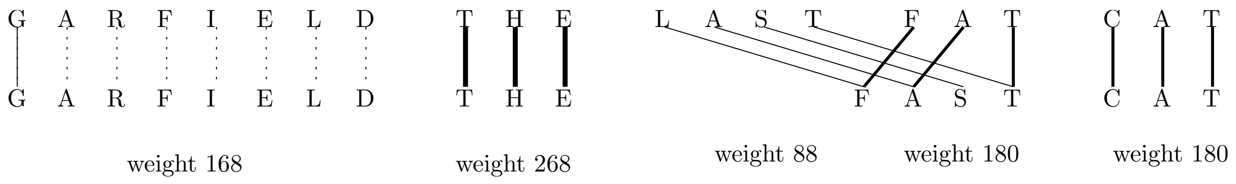
Lets go back to our example. We want to extend the primary library for the pair s_1, s_2. Note that the primary library has uniform weights since we did not combine more than one.

```
a1=GARFIELD THE LAST FAT CAT
   ******** ***  ***  **       weight = 88
a2=GARFIELD THE FAST CAT ---

a1=GARFIELD THE LAST FA-T CAT
   ******** ***      ** * ***
a3=GARFIELD THE VERY FAST CAT  weight = 80
   ******** ***      **** ***
a2=GARFIELD THE ---- FAST CAT

a1=GARFIELD THE LAST FA-T CAT
            ***      ** * ***
a4=         THE      FA-T CAT  weight = 100
            ***      ** * ***
a2=GARFIELD THE      FAST CAT
```

We combine the above extensions into the weighting scheme for the pair of sequences s_1, s_2 (spot the error in the figure !).



Finally we use the above weighting scheme to compute the pairwise alignment using a dynamic programming algorithm without gap penalties. All the information is already incorporated into the libraries. In our case the best pairwise alignment would be:

```
a1=GARFIELD THE LAST FA-T CAT
a2=GARFIELD THE ---- FAST CAT
```

This will correct the error in the inital alignment.

5.6 Computing the progressive alignment

In order to compute the progressive alignment we compute the distance matrix using the extended alignment library as computed above. This again is used to compute a guide tree using the neighbor joining method.

The gaps introduced in the first alignment are fixed and cannot be shifted later.

When aligning two groups of sequences (containing possibly only one sequence) the *average* score from the extended libraries is used for each column.

5.7 Combining TCoffee with the multiple srm

The just presented, original T-Coffee uses *libraries* which are simply alignment graphs where each vertex corresponds to a single residue or nucleotide. Similar to T-Coffee's library, the alignment graph contains alignment information about a set of sequences $\mathcal{S} = \{S^0, S^1, ..., S^{n-1}\}$.

Given such an initial alignment graph, we apply the triplet extension introduced in T-Coffee. Whereas T-Coffee ensures consistency on pairs of characters we ensure consistency on pairs of vertices.

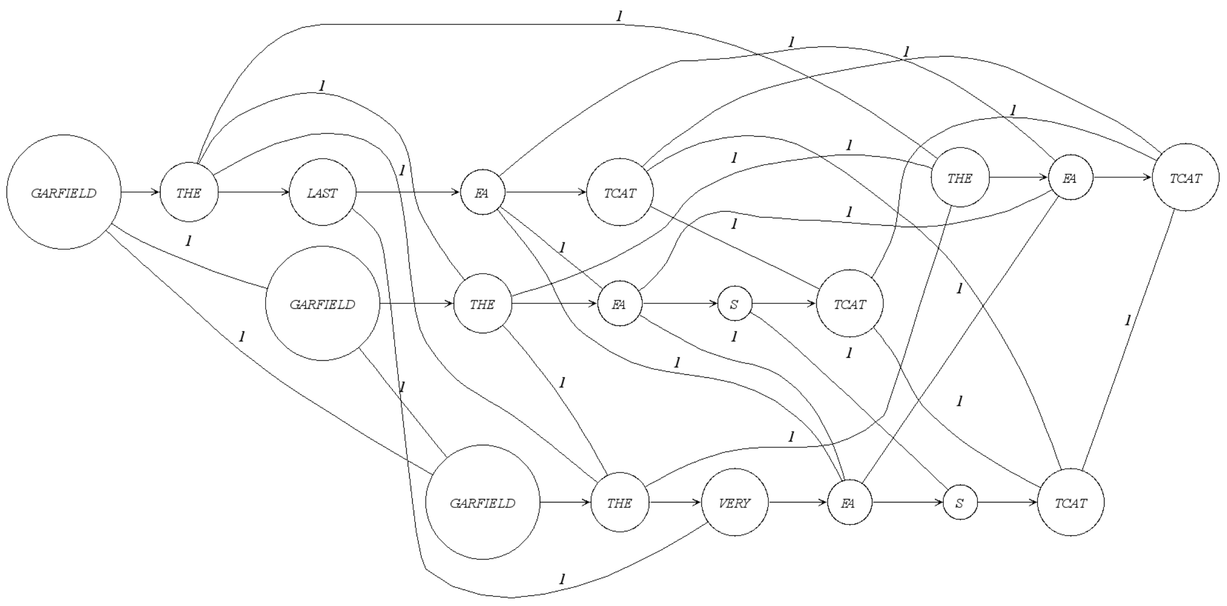
5.8 Combining TCoffee with the multiple srm

Similarly, a progressive alignment is possible on graphs, i.e., a pairwise alignment simply aligns two strings of vertices instead of sequence characters.

A profile in terms of an alignment graph is a string where each position has a set of vertices. Thus, given a guide tree we can perform a progressive alignment on an alignment graph of multiple sequences in the same manner as aligning the sequences themselves.

The approach is, however, more generic because a single vertex can be any group of characters, e.g., a large segment, a gene, or just a single character.

5.9 Combining TCoffee with the multiple srm



5.10 Combining TCoffee with the multiple srm

The advantage of our method is that the input can be *any* set of segment matches. To illustrate it, we explain two different mechanisms to generate segment matches.

For protein alignments and short DNA sequences the standard match generation algorithm takes a set of sequences and builds all pairwise global and local alignments using the dynamic programming algorithms of Gotoh or Waterman and Eggert. We compute the dynamic programming matrix column-wise and save the traceback pointers in a reduced alphabet that can be efficiently stored. This enables us to use these algorithms for fairly long sequences, e.g., a set of adenovirus genomes.

However, for very long (genomic) sequences algorithms using quadratic space are impossible to use and space-efficient dynamic programming algorithms become too inefficient. For these problem instances we either use the enhanced suffix arrays to compute maximal unique matches or external tools such as BLAST.

5.11 Combining TCoffee with the multiple srm

Aligner	= 6	≥ 5	≥ 4	≥ 3	Avg. identity	CPU Time (s)
DIALIGN-T	7888	12161	18187	27690	48%	1259
SeqAn::T-Coffee	12795	18525	25147	32396	63%	1751
MAFFT*	12450	18011	24624	32084	62%	118
MUSCLE*	50	817	5257	21849	38%	673
SeqAn::T-Coffee*	**12911**	**20078**	**27011**	**33147**	**65**%	328

Alignment of 6 adenoviruses: Running time and alignment quality of an alignment of 6 adenoviruses. The number of columns with at least 6, 5, 4, and 3 identical characters are reported together with the average identity.

As it can be seen the combined match refinement and TCoffee approach outperforms the currently best programs in the field.

5.12 Summary

You should know:

- Pairwise and multiple match refinement are efficient methods to preprocess overlapping segment matches.
- In the worst case, segment match refinement can refine a set of input matches to single base matches.
- TCoffee is a consistency based alignment algorithm that computes heuristically a multiple trace.
- The combination of multiple match refinement and TCoffee results in a versatile method to compare sequences based on a set of input segment matches.