Appendix 1: Finding the best structure

The best structure for the prediction is here defined as the nested structure with the highest expected number of correctly predicted positions. This structure is found by a dynamical programming approach similar to the CYK algorithm, using pairing probabilities which can be calculated from the inside-outside variables. Denote the sub-alignment from position i to j as $C_{i,j}$ (i.e. columns i through j in the alignment) and let C_i represents column i by itself. Write the inside and outside variables as (1):

$$\text{Inside:} \qquad e_{i,j}(N) = P(N \rightarrow C_{i,j})$$

$$\text{Outside:} \qquad f_{i,j}(N) = P(S \rightarrow C_{1,i-1} N C_{j+1,n})$$

where N is a non-terminal, S is the starting non-terminal, and n is the alignment length. Assume that all pairs are formed by production rules of the type: $N_1 \rightarrow dN_2d$, where the d's represent a pair. The probability (under the model) that position i pairs with j is:

$$P_d(i,j) = \sum_{N_1,N_2} f_{i,j}(N_1) P(N_1 \rightarrow C_i N_2 C_j) e_{i+1,j-1}(N_2)$$

The sum is effectively only over the rules forming pairs. The probability that position i does not form a pair is:

$$P_s(i) = 1 - \sum_{j \neq i} P_d(i,j)$$

Define $E_{i,j}$ as the maximum possible number of expected correct non-nested predictions in the window from position i to position j in the alignment. Setting $E_{i,j}$ = 0, for $i > j$, the $E_{i,j}$'s can be calculated for $i \leq j$:

$$E_{i,j} = \max\begin{cases} E_{i+1,j} + P_s(i) & \text{(Unpaired)} \\ E_{i+1,j-1} + P_d(i,j) & \text{(Paired)} \\ E_{i,j-1} + P_s(j) & \text{(Unpaired)} \\ E_{i,k} + E_{k+1,j} \quad i \le k \le j & \text{(Bifurcation)} \end{cases}$$

The $E_{i,j}$'s can now be calculated in order of increasing j-i. Actually, the unpaired part of the recursion could be done by the bifurcation step, by initialising the $E_{i,i}$'s to $P_s(i)$, but the above is easier to interpret. $E_{1,n}$ is the maximum expected number of correctly predicted positions in the alignment. The prediction that gives this maximum can be found by backtracking the above recursion.

The reliability value of the prediction in each position is given by the probability that the prediction is correct under the model, i.e. the appropriate P_d or P_s value.

Some non-nested structures may exist which have higher expected numbers of correctly predicted positions than the nested one found above. This could happen even though the SCFG used to find pairing probabilities does not take non-nested structures into account. This will be a rare occurrence and would probably not give any practical information. The structure could be found by a maximal weighted matching approach on the pairing probabilities (2).

References

1. Lari, K. and Young, S. J. (1990) The estimation of stochastic context-free grammars using the inside-outside algorithm. *Comput. Speech Lang.*, **4**, 35-56.
2. Tabaska, J. E., Cary, R. B., Gabow, H. N. and Stormo, G. D. (1998) An RNA folding method capable of identifying pseudoknots and base triples. *Bioinformatics*, **14**, 691-699.

Appendix 2: Implementation details

Calculations

Pfold has two primary time consuming elements. With an alignment length of *n*, they are:

- Doing the inside-outside calculations, which has a running time of $O(n^3)$ (1).
- Calculating the column pair probabilities, which has a running time of $O(n^2)$.

It seems that the inside-outside calculations might consume the most time. It turns out, however, that the constant in the column pair probability calculation can be quite large. This part can be the most time consuming of the calculations when short sequences or many aligned sequences are analysed. The reason is that the column pair probabilities are calculated by multiplying vectors with 16 by 16 matrices in a post-order traversal of the tree (2). For large trees, many such matrix multiplications has to be done.

Time can be saved by avoiding calculations of column pairs with the same nucleotides more than once. This is effective for trees in the mid size range because different columns often have the same nucleotides, e.g. all identical. When analysing larger trees, a few mutations have occurred in most columns often making them slightly different. To cope with this situation, partial column calculations can be stored for later use.

This has been implemented in the following way: when a calculation is made, the result is stored for the sub-tree below each node. When a new calculation is made, the results up to a given node are re-used if the sequence positions are the same for the sequences in the sub-tree below the node. This drastically reduces computation time, since no identical matrix multiplications are done twice. It has the cost of a high memory usage and in some cases a compromise is useful. This was implemented by only saving the results from a sub-tree if it was likely that it would be encountered again. Ideally, this likelihood for nucleotides in a sub-tree should be calculated using the same matrix as in the tree estimation described above. Probabilities for column pairs are closely approximated as the products of the individual columns, since two given columns has a probability of less than $1/n$ of forming a pair and non-pairing columns are treated as independent. Instead of using the matrix from the tree calculations, the matrix for single positions was used, since these calculations are already used in the algorithm.

Speed

For a discussion of the computational speed we use a simulated alignment of 24 sequences of length 1200, related by a balanced unrooted tree with all branch length equal to one expected replacement per ten sites, according to the Jukes-Cantor model (3). Calculations were done on a 700 MHz Intel Pentium III computer with 256 Mb ram.

The finished version of the program can predict the structure of the simulated alignment in about 220 seconds. Without storing partial column probability results in the nodes, the computation time increases by a factor of 1.8. This factor is very dependent on the alignment and phylogeny in question. The factor will be higher for: shorter alignments, more sequences, and for more closely related sequences. For example, the factor is 9.8 for a set of 48 sequences of length 400, related again by a balanced tree, but with branch lengths a tenth of the above.

The time saving by only estimating the phylogeny once is very significant. To shed light on this, we assume that the steps of an ML search will be roughly the same for a structural model and for a model where sites are treated independently. Thus, we can use the number of steps performed by a standard ML method of branch length optimisation. For the test set, a maximum likelihood estimation of the branch lengths required 1787 estimates under the Jukes Cantor model. The ML estimation was done one branch at a time until no significant change in likelihood occurred. This could be done in a more sophisticated way, e.g. by the BFGS method described by Press *et al.* (4). This is unlikely to improve the number of likelihood evaluations to less than 400, since there are 45 branch lengths to estimate. This means that the speed improvement would be of this magnitude for the test set.

References

1. Lari, K. and Young, S. J. (1990) The estimation of stochastic context-free grammars using the inside-outside algorithm. *Comput. Speech Lang.*, **4**, 35-56.

2. Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.*, **17**, 368-376.

3. Jukes, T. H. and Cantor, C. R. (1969) Evolution of protein molecules. In Munro, H. N. (ed.), *Mammalian Protein Metabolism*. Academic Press, New York, pp. 21-132.

4. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992) Nu*merical Recipies in C*, second edition. Cambridge University Press, Cambridge.