Using a β-Contact Predictor to Guide Pairwise Sequence Alignments for Comparative Modelling Supplementary Material

Filippo Ledda[a], Giuliano Armano[a] and Andrew C.R. Martin[a,b]
[a]Dipartimento di Ingegneria Elettrica ed Elettronica,
University of Cagliari, Piazza d'Armi,
I-09123, Cagliari, Italy
[b]Institute of Structural and Molecular Biology,
Division of Biosciences, University College London,
Darwin Building, Gower Street, London WC1E 6BT, UK

15th October, 2010

1 Defining the Cost Function in Detail

Given a pairwise sequence alignment, A, between a template and a target sequence (the structure of the template being known), we define the cost, $c(A, S_{tpl})$, as the sum of three contributions:

$$c(A, S_{tpl}) = c_{nw}(A) + c_{\beta i}(A, S_{tpl}) + c_{bc}(A, S_{tpl}) \tag{1}$$

where S_{tpl} is the structure of the template sequence.

The component c_{nw} is the result of a classical similarity-based scoring scheme with affine gap penalties:

$$c_{nw}(A) = c_{go} \cdot n_{go} + c_{gx} \cdot n_{gx} + \sum_{i,j} M_c(P_{tpl_i}, P_{tgt_j}) \tag{2}$$

where i and j indicate the position of the substituted residues, n_{go} the number of gap openings, and n_{gx} the number of gap extensions, all according to the alignment A. c_{go} is the cost of opening of a new gap, c_{gx} is the cost of extending a gap, and M_c is a substitution cost matrix obtained by reversing a similarity scoring matrix M_s (such as BLOSUM62):

$$M_c = -(M_s - \max(M_s)) \tag{3}$$

The second term, $c_{\beta i}$, in Equation 1 is an additional gap penalty to penalize gaps within the β-strands of the template:

$$c_{\beta i} = n_{g\beta} \cdot c_{g\beta} \tag{4}$$

where $c_{g\beta}$ is β-specific gap penalty and $n_{g\beta}$ is the number of gaps within β-strands.

The last term in Equation 1, c_{bc}, results from the evaluation of β-pairs in the target sequence (assigned from the template, based on the alignment). Considering $\tilde{p}(bp_{tgt})$ as the estimation of the probability of the β-pair bp_{tgt} being formed, it is reasonable that the cost $c(bp_{tgt})$ should be proportional to $-\tilde{p}(bp_{tgt})$. The value of the cost should be large enough to allow the overall cost function to escape from misleading minima obtained with the standard scoring system. Thus, the cost should also be proportional to the number of residues involved in the pairing (n_{bp}). The following quadratic formula gave the best stable performance:

$$c_{ev_{abs}}(bp_{tgt}) = \begin{cases} (0.5 - \tilde{p}(bp_{tgt}))^2 \cdot n_{bp} & \text{if } \tilde{p}(bp_{tgt}) \leq 0.5, \\ -(0.5 - \tilde{p}(bp_{tgt}))^2 \cdot n_{bp} & \text{if } \tilde{p}(bp_{tgt}) > 0.5. \end{cases} \tag{5}$$

An additional term has been included in order to stabilize the algorithm in the presence of wrong estimations. This contribution also takes account of the corresponding β-pair in the template (bp_{tpl}), for which the estimation error is known to be $1 - \tilde{p}(bp_{tpl})$. The requirement for this term derives from the assumption that, with the correct alignment, the errors in the estimation of p on the template and target are correlated. Therefore, if $\tilde{p}$ for the template is significantly larger than for the target, we have a strong indicator of a probable error in the alignment. This relative contribution is given by:

$$c_{ev_{rel}}(bp_{tgt}, bp_{tpl}) = \begin{cases} \left(\tilde{p}(bp_{tpl}) - \tilde{p}(bp_{tgt})\right)^2 \cdot n_{bp} & \text{if } \tilde{p}(bp_{tpl}) - \tilde{p}(bp_{tgt}) > 0.1 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The total cost for a single β-pair is then:

$$c_{ev}(bp_{tgt}, bp_{tpl}) = \gamma_{abs} \cdot c_{ev_{abs}}(bp_{tgt}) + \gamma_{rel} \cdot c_{ev_{rel}}(bp_{tgt}, bp_{tpl}) \tag{7}$$

where γ_{abs} and γ_{rel} are given as parameters. (Note that in NOBCalign, γ_{abs} and γ_{rel} are set to zero.) The total cost for an alignment is given by the sum of c_{ev}, for all the β-pairs resulting from the assignment of the template structure (S_{tpl}) to the target ($\tilde{S}_{tgt}$):

$$c_{bc}(A, S_{tpl}) = \sum c_{ev}(bp_{tgt}, bp_{tpl}), \forall bp_{tpl} \in S_{tpl}, bp_{tgt} \in \tilde{S}_{tgt} \tag{8}$$

```
def bc_align(template_seq, target_seq, template_str):
    c_pairs=set()
    for MAX_ITERATIONS times:
       current_solution = bc_search(template_seq, target_seq, c_pairs)
       new_c_pairs = c_pairs + get_c_pairs(current_solution, template_str)
       if len(new_c_pairs) > len(c_pairs):#no new pair found
           c_pairs = new_c_pairs
       else:
           break
    return current_solution
```

Figure S1: The iterative algorithm pseudo-code, in Python-like syntax.

2 Minimizing the Cost Function In Detail

A search algorithm is completely defined by:

- *The search problem.* This is defined by the tuple:
  (S_0, operator-set, goal-test, f),
  where S_0 is the start state; *operator-set* defines the set of states that can be reached from a given state; *goal-test* can say whether a given state is the goal or not; f is an evaluation function which gives a score (or cost) for a given path (sequence of states).

- *The search strategy.* This determines the order in which the nodes are expanded. For instance, a best-first strategy always expands a node with the best value of f.

With a global-search algorithm, the best alignment can be found by searching for the path in a tree which optimizes f, leading to the end of the sequences.

Using a blind (brute-force) search strategy, many nodes are expanded unnecessarily before finding the solution; this may lead to an explosion in computational cost. The number of expanded nodes is greatly reduced by adopting a heuristic search algorithm such as A* where the path cost of a given node n is the sum of two terms:

$$f(n) = g(n) + h(n) \tag{9}$$

g being a path cost function, and h being a heuristic function, expected to estimate the cost from that node to the solution. If the cost increases monotonically along the path and the heuristic function is 'admissible' (i.e. it is an underestimation of the real cost of the solution) the A* algorithm is guaranteed to find the path with minimum cost. The complexity becomes linear if the estimation given by the heuristic function is exact.

Using the cost function defined in Equation 1 as $g(n)$, a heuristic function can be devised to estimate exactly the first two terms of the cost, but an exact

estimate cannot be given for $c_{bc}(A, S_{tpl})$; the complexity rapidly increases with the length of the sequences to be aligned. In order to control the run time, rather than dynamically apply the evaluator during the search, an iterative procedure has been used: at the *i-th* iteration, the pairings found in all $i-1$ previous iterations are evaluated, until a reasonable trade-off is found.

Figure S1 presents the pseudo code for the iterative procedure. The function *bc_search* performs a search, applying the costs for the pairs obtained for the β-pairs encountered in the solutions of previous iterations. The iterations continue until convergence, or until an iteration limit is reached.

The A* evaluation function has been used in the alignment search algorithm, with the Iterative-Deepening A* (IDA*) search strategy. In our case, a search state is indicated by $S = [i, j]$. i and j represent the relative position in the sequences, and can also be viewed as coordinates in a Needleman and Wunsch-like cost matrix. The heuristic search problem is given by:

- S_0 **(start state):** $[0, 0]$
- **goal-test:** $[i_0, j_0], i_0 = length(P_1) \wedge j_0 = length(P_2)$
- **operators:** $[i_0, j_0] \rightarrow \{[i_0 + 1, j_0 + 1], [i_0 + 1, j_0], [i_0, j_0 + 1]\}$ (substitution, insertion, and deletion respectively). Each operation is defined provided that $i \leq \ length(P_1) \vee j \leq length(P_2)$.
- g **(cost function):** depends on the path from the start state to the current state. It includes the costs for the substitutions and gaps along the path, and the costs arising from β-pair evaluations. The cost function is basically c as defined in Equation 1, but for performance reasons, the values are rounded to the nearest integer.
- h **(heuristic function):** the heuristic function is the estimated cost for the remaining part of the sequences from the current state. A matrix H_{P_1,P_2} gives the estimation: $h([i, j]) = H_{P_1,P_2}(i, j)$ and is constructed similarly to a Needleman and Wunsch matrix, minimizing the sum of the terms c_{nw} and $c_{\beta i}$.

3 Results

Fold	Accuracy	Precision	Recall	MCC†
1	0.781	0.765	0.807	0.563
2	0.783	0.774	0.797	0.565
3	0.795	0.777	0.821	0.592
4	0.784	0.777	0.802	0.568
5	0.784	0.768	0.812	0.569
6	0.790	0.764	0.840	0.582
7	0.778	0.770	0.797	0.556
AVG	0.785	0.771	0.811	0.571

Table S1: Results for BCeval in a 7-fold cross validation test on the dataset TRAINCH. †MCC = Matthews' Correlation Coefficient.