Pseudo-Boolean Multiple Sequence Alignment

Steven Prestwich
s.prestwich@cs.ucc.ie
Cork Constraint Computation Centre
Department of Computer Science

Des Higgins and **Orla O'Sullivan**
d.higgins@ucc.ie
ojos@student.ucc.ie
Department of Biochemistry

University College, Cork, Ireland

Background

Multiple sequence alignment (MSA) is a central problem in Bioinformatics and is known to be NP-complete. Given a number of sequences of symbols from an alphabet, the aim is to build an *alignment matrix* that maximises some function. Gaps may be introduced between symbols, and in some MSA formulations the objective function includes a measure of the number and length of gaps. Numerous heuristic methods have been proposed for multiple alignment, of which by far the most widely used is *progressive alignment*. This involves clustering the sequences first to give a guide tree and then building up the alignment gradually, following the branching order in the guide tree. This is very fast even for hundreds of sequences, and the most widely used software is the well-known ClustalW package. The T-Coffee package also uses a progressive heuristic but has been shown to be more accurate than ClustalW, at the expense of extra computing time.

The application of *generic* approaches to the MSA is of interest because (i) it tests our search and modelling tools, (ii) it may yield good results, (iii) it allows us to experiment with alternative models. Dynamic programming has been used for MSA problems but is known to scale poorly to more than a few sequences. There are several methods based on optimising the *weighted sums of pairs* objective function which use Genetic Algorithms or iteration. These vary in the extent to which they are practical for more than a few sequences or in the quality of the optimisation. Another approach is the *Complete Maximum Weight Trace* (CMWT) formulation in which the symbols are viewed as vertices in an *alignment graph* $G = (V, E)$. Each vertex is a position in a sequence. Each edge connects two vertices from different sequences and each edge has a *weight* that estimates the usefulness of aligning its two symbols. An alignment *realises* an edge if it aligns its two symbols, and the aim is to maximize the sum of the weights of the realised edges. The set of realised edges is a *trace* if certain constraints are satisfied, and an alignment matrix can always be constructed from a trace.

The CMWT generates large models which can be reduced by using the *Sparse Maximum Weight Trace* (MWT) formulation. This restricts attention to a carefully chosen subgraph, defining only those edges that are used in pairwise alignments of high quality. Besides reducing the size of the models, the MWT provides an opportunity to input biological knowledge via the choice of subgraph. The usual way of ensuring that the realised edges form a valid trace is to enumerate all *critical mixed cycles* in the graph, adding a constraint to prohibit each cycle (other constraints may also be added). The MWT and related formulations have natural integer linear

program (ILP) models. The number of constraints is exponential in the size of the graph but these are not passed en masse to a solver. Instead a branch-and-cut approach is used, generating violated constraints as required in order to derive cutting planes. Generating the relevant constraints is known as the *separation problem* and can be done in polynomial time.

Our approach

We propose an alternative ILP model. This is transformed to linear pseudo-Boolean (PB) form and passed to the Saturn hybrid local search algorithm for solution. PB is a generalization of SAT which significantly improves expressiveness, yet many SAT algorithms generalise quite easily to PB. Simple post-search transformations are applied to the matrix in order to remove superfluous columns.

The model is of polynomial size, and therefore better suited to a generic solver such as Saturn. As in the MWT a binary variable corresponds to each graph edge. To avoid the generation of exponentially many cycle constraints the alignment matrix is modelled directly, and to allow the modelling of reasonably large problems a *log encoding* is used to specify the matrix column corresponding to each sequence position. Ordering constraints are added to preserve the sequences, and the matrix and graph variables are related by channelling constraints.

Two sets of implied constraints are added: additional ordering constraints on non-adjacent symbols, and stronger channelling constraints exploiting the fact that any integer has a unique binary representation. Other implied constraints were tried, including small cycle constraints, but these did not help. Various forms of symmetry breaking are possible, but we do not use them because adding such constraints can harm local search performance, though they usually aid backtrack search. Interestingly, a class of transitivity constraints are not implied in our model and are a form of symmetry breaking, but as an optimal alignment is approached they become implied. In tests they were not helpful, but they became less harmful as optimality was approached.

Results

We took several MSA instances from the HOMSTRAD database of protein alignments. For each instance we generate the sparse alignment graph using T-Coffee with default settings. We usually found the known best alignments on the smallest instances so we tried two larger ones. The mmp problem is a family of matrix metalloproteineases which are important proteins in the cytoskeleton of the cell. It has 6 sequences and an average length of 164 residues. The alignment found by Saturn equals that found by T-Coffee and beats that found by ClustalW. The oxidored_q6 problem is a family of NADH ubiquinone oxidoreductases which are enzymatic proteins involved in the Citric Acid Cycle in the cell. It has 5 sequences and an average length of 265 residues. The alignment found by Saturn beats those found by both ClustalW and T-Coffee. Thus on at least some problems the PB approach is competitive with state-of-the-art systems in terms of solution quality. It is far slower, taking tens of minutes as opposed to seconds or less, and is unlikely to scale to truly large problems; but these are promising first results and indicate the form that a much faster specialised version of the algorithm should take. In future work we aim to implement such an algorithm, avoiding the use of the (often harmful) log encoding, and possibly enhancing performance by borrowing techniques from the branch-and-cut approach.