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

Abstract

Multiple sequence alignment is a central problem in Bioinformatics and a challenging one for optimisation algorithms. An established integer programming approach is to apply branch-and-cut to a graph-theoretical model. The models are exponentially large but are represented intensionally, and violated constraints can be located in polynomial time. This report describes a new integer program formulation that generates polynomial-sized models small enough to be passed to generic solvers. It is a hybrid formulation relating the sparse alignment graph with a compact encoding of the alignment matrix via channelling constraints. Alignments obtained with a pseudo-Boolean local search algorithm are competitive with those of state-of-the-art algorithms. Execution times are much longer, but in future work we aim to develop a more efficient specialised algorithm.

1 Introduction

Multiple sequence alignment (MSA) is a central problem in Bioinformatics and is known to be NP-complete [15]. Given a number of sequences of symbols from an alphabet, the aim is to align them while maximizing 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. A common data structure is the *alignment matrix* which contains one sequence per row, including gaps; aligned symbols occur in the same column.

Numerous heuristic methods have been proposed for multiple alignment, of which by far the most widely used is so-called 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 [28]. The T-Coffee package [20] also uses a progressive heuristic but has been shown to be more accurate than ClustalW, at the expense of

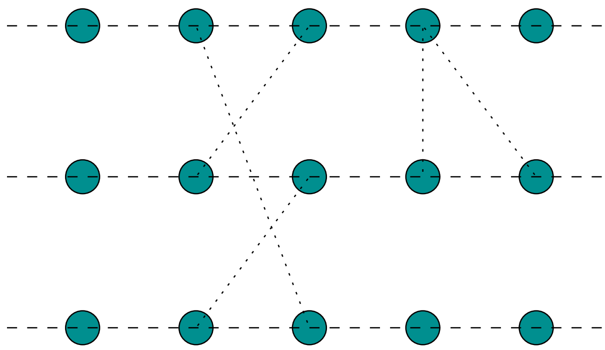

Figure 1: An alignment graph with small cycles

extra computing time. There are also several methods based on optimising the WSP (weighted sums of pairs) objective function which use Genetic Algorithms [19] or iteration [11]. These vary in the extent to which they are practical for more than a few sequences or in the quality of the optimisation.

Dynamic programming [18] has been used for MSA problems but is known to scale poorly to more than a few sequences. More successful is the *Complete Maximum Weight Trace* (CMWT) formulation in which the symbols are viewed as vertices in an *alignment graph* $G = (V, E)$ (actually an *extended* graph that includes edges between adjacent symbols in each sequence). Each vertex is a position i in sequence j, which we shall denote by (i, j). Each edge connects two vertices from different sequences: we shall denote an edge $e \in E$ between (i, j) and (i', j') by $e = ((i, j), (i', j'))$ where $j < j'$ by convention. Each edge $e \in E$ has a *weight* w_e representing 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 [1, 16, 24] (other constraints may also be added). Figure 1 shows examples of cycles that prevent the trace from representing a alignment matrix. The MWT and related formulations have natural integer linear program (ILP) models. The number of constraints is exponential in the size $|E|$ of the graph [1] 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.

In this report we explore an alternative ILP approach to the MSA. Instead of accessing an exponentially large model in polynomial time, we use a model that is only polynomial in size and can be passed to a generic ILP solver. To avoid the use of cycle constraints we model the alignment matrix directly, and relate it to the sparse alignment graph by *channelling constraints* [5]. The Constraint Programming literature contains many examples of problems in which extra variables are introduced to facilitate the expression of certain constraints, which must be related to the original variables via channelling constraints. The motivation is that some constraints can be more easily or economically expressed on different sets of variables. Introducing extra variables is not necessarily counter-productive, and hybrid models sometimes give better results than those with fewer variables. Given such a model, we then face the choice of how to find solutions. Instead of applying branch-and-cut we transform the model to pseudo-Boolean form and pass it to a local search algorithm, an approach that has given good results on a variety of combinatorial problems. We also exploit modelling techniques aimed specifically at local search.

The report is organised as follows. Section 2 describes the model, Section 3 describes the techniques used to solve the problem, Section 4 evaluates the usefulness of the model on examples, and Section 5 draws conclusions and discusses future work. A condensed version of this report in published in [23].

2 A hybrid ILP model

This section describes the new ILP model of the MSA, drawing on techniques from the Constraint, Satisfiability and Integer Programming literatures. Section 2.1 describes an initial model, Section 2.2 discusses symmetry breaking, Section 2.3 adds implied constraints to improve constraint propagation, and Section 2.4 discusses properties of the model.

2.1 Initial model

As in the MWT define a binary variable v_e for each edge $e \in E$. The problem is then:

$$\text{maximize} \sum_{e \in E} w_e v_e$$

subject to constraints ensuring the construction of an alignment matrix. The matrix has n rows (one per sequence) and c columns where $c \geq \max_j(l_j)$ and l_j is the length of sequence j. We allow each sequence position to be placed anywhere in the corresponding row of the matrix, subject to constraints. A matrix entry with no sequence position placed in it implicitly contains a gap. The symbols are not explicitly modelled, only the way in which sequence positions are mapped to matrix columns: define an integer variable p_{ij} with domain $\{1, \ldots, c\}$ for each sequence position (i, j) denoting its matrix position (p_{ij}, j). We need only define p_{ij} for sequence positions that lie on an edge of the sparse alignment graph, but we shall assume that they all do.

We require a 0/1 ILP model so we next replace the p_{ij} by 0/1 variables. To enable the modelling of long sequences we use a *log encoding* by modelling binary representations of the p_{ij}. Several researchers have used log encodings to model constraint satisfaction problems (CSPs) as SAT (propositional satisfiability) and found that both local and backtrack search performance is usually worse than on other encodings [8, 9, 12], and [30] has shown that propagation is weaker on log encodings. Nevertheless, this type of model seems worth trying because of its compactness. Let $b = \lceil \log_2 c \rceil$ so that the c matrix columns can be represented using b bits. Define 0/1 variables p_{ijk} where $1 \leq j \leq n$, $1 \leq i \leq l_j$ and $1 \leq k \leq b$, such that

$$p_{ij} = 1 + \sum_{k=1}^{b} 2^{k-1} p_{ijk}$$

That is p_{ijk} is the k^{th} bit of the binary representation of $p_{ij} - 1$.

The initial model has two sets of constraints, expressed in 0/1 form by substituting the above formula for p_{ij}. To ensure that sequence positions are placed in the alignment matrix in an ordered way, add *ordering constraints*

$$p_{ij} - p_{i-1,j} \geq 1$$

where $1 \leq j \leq n$ and $1 \leq i \leq l_j$. To relate the p_{ij} and v_e variables add *channelling constraints*

$$v_e = 1 \rightarrow p_{ij} = p_{i'j'}$$

where $e = ((i, j), (i', j')) \in E$. Each channelling constraint can be implemented by two linear constraints

$$p_{ij} - p_{i'j'} + cv_e \leq c \qquad p_{i'j'} - p_{ij} + cv_e \leq c$$

That is, if $v_e = 1$ then p_{ij} is both $\leq$ and $\geq$ the sum of $p_{i'j'}$. If $v_e = 0$ then both constraints are trivially true. Note that the channelling constraints only work one way, so that $p_{ij} = p_{i'j'} \not\rightarrow v_e = 1$. The reason is discussed in Section 2.2.

2.2 Symmetry breaking

Many combinatorial problems exhibit symmetries, for example the N-queens problem has eight: each solution may be rotated and reflected. Removing symmetries from a problem can reduce its search space by a large factor, greatly reducing the time to find a solution and to prove it optimal. Probably the most popular approach is to add constraints to the model, so that each equivalence class of symmetric solutions to the original problem corresponds to a single solution in the new problem.

An obvious symmetry in our model involves gaps in the alignment matrix. Any MSA solution may have many symmetric solutions in which gaps are inserted between columns. These can be avoided by adding constraints to restrict the width of the alignment matrix. A less obvious form of symmetry involves the relationship between the p_{ijk} and v_e variables. An assignment $v_e = 1$ implies that the sequence positions connected by edge e are aligned in the matrix, but the reverse is not true: $v_e = 0$ does not imply that the two sequence positions connected by e are unaligned. Thus any solution in which two symbols on an edge e are unaligned has two symmetric solutions: with $v_e = 1$ and $v_e = 0$ (unless the latter case would cause the total weight to be less than a lower bound). This symmetry can be removed by adding constraints to enforce the reverse implication, though this probably requires either a large number of linear constraints or the introduction of further variables.

However, recent work [21] has shown that adding symmetry breaking constraints to a model is often counter-productive when a local search algorithm is to be applied. For local search performance the size of the search space appears to be less important than the number and distribution of solutions in the space, and symmetry appears to be harmless. We therefore do not break symmetries in our model.

2.3 Implied constraints

The initial model ensures a correct alignment, but it can be improved by adding *implied constraints* — constraints that are implied by others already in the model. These have been shown to speed up both backtracking [27] and local search [4, 14, 21].

2.3.1 Stronger ordering constraints

A larger set of ordering constraints can replace those of the initial model:

$$\sum_{k=1}^{b} 2^{k-1}(p_{ijk} - p_{i'jk}) \geq i - i'$$

where $1 \leq j \leq n$ and $1 \leq i' < i \leq l_j$. Similar constraints were found to improve local search performance on SAT-encoded clique problems [21], and they also help on this model.

2.3.2 Stronger channelling constraints

The channelling constraints can also be strengthened. Every integer has a unique binary representation, so $p_{ij} = p_{i'j'}$ if and only if $p_{ijk} = p_{i'j'k}$ where $1 \leq k \leq b$. Therefore the following constraints can be used:

$$p_{ijk} - p_{i'j'k} + v_e \leq 1 \qquad p_{i'j'k} - p_{ijk} + v_e \leq 1$$

where $1 \leq k \leq b$ and $e = ((i,j),(i',j')) \in E$. Given an assignment $v_e = 1$ a constraint propagation algorithm can then remove a value from the domain of any bit p_{ijk} $[p_{i'j'k}]$ for which the corresponding bit $p_{i'j'k}$ $[p_{ijk}]$ is currently assigned. Under the weaker channelling constraints, domain values can only be pruned on the larger p_{ijk} $[p_{i'j'k}]$ when several of the $p_{i'j'k}$ $[p_{ijk}]$ have been assigned.

2.3.3 Positioning constraints

We can restrict the range of each p_{ij} by adding constraints

$$i \leq p_{ij} \leq i + c - l_j$$

for $1 \leq j \leq n$ and $1 \leq i \leq l_j$. In any correct alignment the p_{ij} will always lie within these ranges. In tests these constraints made little difference, perhaps because the stronger ordering constraints provide similar information.

2.3.4 Small cycle constraints

Though the MWT constraints are unnecessary in our model, we can use some of them as implied constraints. For example to prevent cycles involving 2 sequences we could add SAT-style constraints to enumerate pairs of crossing edges:

$$(1 - v_e) + (1 - v_{e'}) \geq 1$$

where $e = ((i,j),(i',j')) \in E$, $e' = ((k,j),(k',j')) \in E$, and either $i \leq k$ and $i' \geq k'$ or $i \geq k$ and $i' \leq k'$. ILP allows a more compact representation:

$$|C_e|\, v_e \leq \sum_{e' \in C_e} v_{e'}$$

where $e \in E$ and C_e is the set of edges crossing e (as defined above). These constraints were also found to have little effect.

2.3.5 Transitivity constraints

In [1] transitivity constraints are used, based on the observation that if symbols A and B are aligned in the matrix, and if B and C are aligned, then so are A and C. It therefore seems reasonable to add transitivity constraints to our model:

$$v_e + v_{e'} \leq 1 + v_{e''} \qquad v_e + v_{e''} \leq 1 + v_{e'} \qquad v_{e'} + v_{e''} \leq 1 + v_e$$

where $e = ((i,j),(i',j')) \in E$, $e' = ((i',j'),(i'',j'')) \in E$ and $e'' = ((i,j),(i'',j'')) \in E$. However, in our model transitivity on the v_e is *not* implied because of the one-way nature of the channelling constraints: the assignment $v_e = 1$ forces the symbols on edge e to be aligned but $v_e = 0$ does not force them to be unaligned.

Thus the transitivity constraints are symmetry breaking constraints in our model, not implied constraints. The distinction is unimportant if backtracking is to be applied because both types of constraint are usually beneficial. But if local search is to be applied then the former are probably harmful and the latter beneficial. Interestingly, the following theorem holds.

Theorem. *In an optimal solution the transitivity constraints on the v_e are implied.*

Proof. Suppose that in an optimal solution $v_e = v_{e'} = 1$ where symbols A and B occur on edge e and symbols B and C on edge e'. Then we know that the pairs A,B and B,C are aligned in the matrix, and therefore so are A,C. This means that $v_{e''}$ can be set to either 0 or 1, where A and C are on edge e''. If $v_{e''} = 0$ the solution is less than optimal, contradicting the initial assumption. Hence $v_{e''} = 1$ and transitivity holds on the v_e. QED.

Such constraints could be called *optimality-implied* and to our knowledge they have not been described previously. As optimality is approached they undergo a transition from being symmetry breaking constraints to implied constraints, so for local search we might expect them to switch from being harmful to helpful. This would be an interesting result, analogous to the fact that different models can be best for finding an optimal solution and for proving it optimal by backtrack search [26]. In experiments they seemed to be harmful but less so as alignment weight increased, and we do not use them in the model.

2.4 Discussion of the model

We call this the Sparse Graph Log Matrix (SGLM) model, indicating its combination of the sparse alignment graph with a log encoding of the alignment matrix. It contains only the stronger ordering and channelling constraints from Section 2.3. Both the SGLM and the MWT have $O(|E|) = O(n^2 m)$ variables v_e; the SGLM has in addition $O(nm \log m)$ variables p_{ijk} which is not prohibitive. A motivation for the SGLM was to avoid the exponential number of constraints in the MWT so we analyse its space complexity. It has $O(nm^2)$ ordering constraints and $O(|E| \log w)$ channelling constraints, where w is the matrix width, n is the number of sequences and m is the sequence length (we assume that the sequences are of similar length). In practice $|E| = O(n^2 m)$ [24] and $w = km$ for some constant $k > 1$, so the model has $O(nm(m + n \log m))$ constraints. Perhaps a better measure of model size is the total number of literals, that is $\sum_i \sigma_i$ where i ranges over the set of constraints and σ_i is the number of literals in constraint i. The SGLM has $O(nm^2)$ ordering constraints of size $O(\log m)$ and $O(n^2 m \log m)$ channelling constraints of size 3, giving a total model size of $O(nm(n+m) \log m)$. Typically $m \gg n$ so this reduces to $O(nm^2 \log m)$ literals.

Even if the transitivity and 2-cycle constraints were found to be useful, they greatly increase the model size and are best avoided: there are $O(|E|^3) = O(n^6 m^3)$ transitivity constraints, and though there are only $O(nm^2)$ 2-cycle constraints they are of size $O(m^2)$, so they contain $O(nm^4)$ literals. It is fortunate that these constraints seem to contribute little to local search performance. However, the stronger ordering and channelling constraints do greatly improve performance, so we use them even though they increase the model size: the initial model with weaker ordering and channelling constraints has only $O(n^2 m \log m)$ literals.

3 Solving the optimisation problem

We reduce an MSA optimisation problem to a series of CSPs, each with a cost constraint

$$\sum_{e \in E} w_e v_e > W$$

for some integer lower bound W. The CSPs have increasing values of W, each being the weight of the previous solution, and the initial value of W is set to 0. We now describe the algorithm used to solve each CSP.

3.1 Pseudo-Boolean local search

ILP contain constraints of the form

$$\sum_i w_i v_i \sim d$$

where $\sim$ is one of the operators $\{<, >, \leq, \geq, =\}$, the weights w_i and the constant d are (possibly negative) integers, and the variables v_i have domain $\{0, 1\}$. Other integer values can be permitted for the v_i but such problems can always be transformed into 0/1 form. 0/1 ILP models can in turn be transformed into *linear pseudo-Boolean* form, involving only positive d and w_i, and literals l which are either variables v or their negations $\bar{v} = 1 - v$. The interest of this form is that it is only a slight extension of SAT, and many SAT algorithms generalise easily to it. This makes sense when trying to solve problems with a strong logical content, which are sometimes better solved by SAT approaches than by cost-based methods. The well-known Davis-Logemann-Loveland-Putnam SAT backtracking algorithm [6] was extended to pseudo-Boolean problems in [2, 3] and applied to integer programming benchmarks and randomly-generated problems. The Walksat local search algorithm for SAT [25] was generalised in [29] to WSAT(OIP) and applied to radar surveillance benchmarks and the Progressive Party problem. The Saturn hybrid local search algorithm was generalised in [22] and applied to hardware verification, block design and a sports scheduling problem. In each case good results were obtained.

We apply Saturn to the SGLM. Saturn performs local search in a space of partial variable assignments that are consistent under constraint propagation. On SAT problems the propagation algorithm used is *unit propagation* which generalises naturally to pseudo-Boolean problems (see [2, 3, 22] for details). The aim is to combine the scalability of local search with the pruning ability of constraint-based algorithms. Like most local search algorithms, Saturn has a runtime noise parameter that must be tuned to each problem or problem class. For this work we modified Saturn slightly. Each CSP can in principle be solved independently; instead we use each solution as a starting point for the next CSP, by assigning as many variables as possible under a random variable ordering, without violating any constraints. This technique was not previously used with Saturn but greatly speeds up convergence on MSA instances. Best results were obtained by executing Saturn in a rapid-restart mode with variable noise. On failing to find a solution of a given weight within a short time, it is restarted from the previous solution with slightly higher noise. On finding a solution the noise is reduced again. This strategy allows Saturn to make many small improvements in a short time, and on becoming trapped in a local minimum it increases its noise in order to escape.

3.2 Post-processing the matrix

On solving the optimisation problem an alignment matrix is constructed, then post-processed by applying simple hill-climbing and plateau traversal techniques. This improves the matrix weight significantly and is done in three phases:

- Randomly-chosen symbols are moved randomly one position to the left or right, where this is possible and does not decrease the *score* of the matrix; score is defined here as a lexicographical ordering on total weight then number of non-weighted symbol matches.
- Symbols are moved randomly one position to the left as many times as possible, where this does not decrease the score.
- If two adjacent columns i and $i+1$ contain no symbols in the same row then the two columns can be merged by shifting the rightmost symbols one place to the left (this includes the case of an empty column i). This phase eliminates unnecessary gaps in the matrix without decreasing the weight, and possibly increasing it.

If Saturn fails to solve a CSP in the allotted time, it is restarted from the best solution so far, after feeding back the matrix improvements into the solution.

4 Experiments

We take MSA instances from the HOMSTRAD [17] database of protein alignments. These are generated by considering the three-dimensional structure of sets of proteins and provide very accurate test cases. We measure the accuracy of our alignments by simply counting the percentage of columns in our alignment matrix that are identical with the test cases. For each test case we generate the sparse alignment graph using T-Coffee with default settings. This takes every pair of sequences and outputs weighted pairs of symbols. These are produced by aligning each pair of sequences using dynamic programming and recording all of the pairs of aligned symbols. The weights are simply the percent identity of the parent sequences for each pair.

We first applied Saturn to four fairly small problems: ChtBD is a family of chitin binding domains which are structural proteins in plant cells, hla consists of a group of histocompatibility proteins involved in the immune system, TIG contains a group of glucanotransferases which are involved in metabolism, and ch is a family of calponin homology domain proteins which are involved in actin binding in the cell. These have between 4 and 6 similar sequences of between 43 and 178 residues, apart from ch which has 4 very dissimilar sequences. In each case Saturn finds the same solutions as ClustalW and T-Coffee in a few minutes, except for ch where it finds a less optimal solution: only 57.9% correct, while ClustalW's solution is 61.7% correct.

Next we consider two larger problems, starting with the mmp problem, a family of matrix metalloproteineases which are important proteins in the cytoskeleton of the cell. This has 6 sequences, 55% identical to each other, and an average length of 164 residues. The SGLM model has 6723 v_e variables, 7872 p_{ijk} variables, 80311 ordering constraints and 107569 channelling constraints. The alignment found by Saturn is shown in Figure 2. It is 95.5% correct, equalling that found by T-Coffee, while ClustalW's alignment is 92.4% correct. Secondly we take the oxidored_q6 problem, a family of NADH ubiquinone oxidoreductases which are enzymatic proteins involved in the Citric Acid Cycle in the cell. This has 5 sequences, 57% identical to each other, and an average length of 265 residues. The SGLM model has 4563 v_e variables, 11934 p_{ijk} variables, 175237 ordering constraints and 82135 channelling constraints. The alignment found by Saturn is shown in Figure 3. It is 98.7% correct, ClustalW's is 97.3%, and T-Coffee's 95.5%. Thus on these problems the pseudo-Boolean approach is competitive with ClustalW and T-Coffee in terms of solution quality. It is far slower, taking tens of minutes as opposed to seconds or less, but these are promising first results.

To confirm empirically that SGLM models are significantly smaller than MWT models, we generated constraints required to prevent critical mixed cycles involving exactly 3 edges. On the mmp problem there are a quarter of a billion such constraints, so the MWT has at least several hundred times more constraints than the SGLM.

5 Conclusion

The multiple sequence alignment problem is interesting from a constraint modelling point of view. The first aim of this work was to obtain a much smaller ILP model than previously described, which could potentially be passed to any ILP solver. This was done by avoiding the usual graph-theoretical formulation, instead using channelling constraints to link the sparse alignment graph with a log encoding of the alignment matrix. We showed theoretically and empirically that the models are much smaller than those obtained by previous ILP formulations designed for branch-and-cut. A variety of possible implied and symmetry-breaking constraints can be used, and we discovered a class of intermediate constraints whose status depends on closeness to optimality. Though these *optimality-implied* constraints turned out to be unhelpful, they may be useful on other problems. The second aim of the work was to evaluate the usefulness of the model by applying a search algorithm. We obtained promising results with a pseudo-Boolean local search algorithm, in one case finding a better result than two state-of-the-art alignment algorithms.

An advantage of the ILP approach is that it makes it relatively easy to add extra constraints expressing special cases or domain knowledge. This allows quick experimentation with alternative weight functions, or the enforcement of desirable alignments. However, it is not competitive with standard alignment software in terms of execution time, though it may be useful on pathological instances that mislead standard heuristics. It is also unlikely to scale up to truly large problems involving hundreds of sequences with thousands of symbols, which specialised algorithms are able to solve. To address these problems we are exploring two directions for future research. Firstly, the local search algorithm could be specialised to sequence alignment. A similar approach to graph colouring and block design gave more than 2 orders of magnitude speed-up, could handle larger problems and gave better results. A specialised algorithm could also exploit cost-based reasoning and would not need to resort to a log encoding. Secondly, the new model could be abandoned, and instead the pseudo-Boolean search algorithm could be extended to handle *lifted* models. These contain constraints expressed intensionally and generated as required, as in branch-and-cut algorithms. The extended algorithm could then be applied to the usual maximum weight trace model. Lifted constraint solvers have already been implemented [10, 13] and extension to Pseudo-Boolean and lifted models are predicted as important near-term SAT advances [7].

Acknowledgment

The Cork Constraint Computation Centre is supported by Science Foundation Ireland. Orla O'Sullivan was paid from a Basic Research grant from Enterprise Ireland to D. Higgins.

References

[1] E. Althaus, A. Caprara, H.-P. Lenhof, K. Reinert. Multiple Sequence Alignment With Arbitrary Gap Costs: Computing an Optimal Solution Using Polyhedral Combinatorics. *Bioinformatics* Suppl 2:S4–S16.

[2] F. A. Aloul, A. Ramani, I. Markov, K. Sakallah. PBS: a Backtrack-Search Pseudo-Boolean Solver and Optimizer. *Fifth International Symposium on Theory and Applications of Satisfiability Testing*, Cincinnati, Ohio, USA, 2002, pp. 346–353.

[3] P. Barth. A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Research Report mpi-i-95-2-003, Max-Plank Institut fur Informatik, Saarbrucken 1995.

[4] B. Cha, K. Iwama. Adding New Clauses for Faster Local Search. *Fourteenth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence 1996, pp. 332–337.

[5] B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, J. C. K. Wu. Increasing Constraint Propagation by Redundant Modeling: an Experience Report. *Constraints* vol. 4, 1999, pp. 167–192.

[6] M. Davis, G. Logemann, D. Loveland. A Machine Program for Theorem Proving. *Communications of the ACM* vol. 5, 1962, pp. 394–397.

[7] H. E. Dixon, M. L. Ginsberg, A. J. Parkes. Likely Near-term Advances in SAT Solvers. *Workshop on Microprocessor Test and Verification*, Austin, Texas, USA, 2002.

[8] A. Frisch, T. Peugniez. Solving Non-Boolean Satisfiability Problems with Stochastic Local Search. *Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, Washington, USA, 2001.

[9] A. van Gelder. Another Look at Graph Coloring via Propositional Satisfiability. *Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, NY, 2002, pp. 48–54.

[10] M. L. Ginsberg, A. J. Parkes. Satisfiability Algorithms and Finite Quantification. *Seventh International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, Colorado, USA, 2000.

[11] O. Gotoh. Significant Improvement in Accuracy of Multiple Protein Sequence Alignments by Iterative Refinement as Assessed by Reference to Structural Alignments. *Journal of Molecular Biology* vol. 264, 1996, pp. 823–838.

[12] H. Hoos. SAT-Encodings, Search Space Structure, and Local Search Performance. *Sixteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1999, pp. 296–302.

[13] D. Joslin, A. Roy. Exploiting Symmetry in Lifted CSPs. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, American Association for Artificial Intelligence 1997, pp. 197–203.

[14] K. Kask, R. Dechter. GSAT and Local Consistency. *Fourteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann 1995, pp. 616–622.

[15] J. D. Kececioglu. Exact and Approximation Algorithms for DNA Sequence Reconstruction. PhD thesis, University of Arizona, 1991.

[16] J. D. Kececioglu, H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, M. Vingron. A Polyhedral Approach to Sequence Alignment Problems. *Discrete Applied Mathematics* vol. 104, 2000, pp. 143–186.

[17] K. Mizuguchi, C. M. Deane, T. L. Blundell, J. P. Overington. HOMSTRAD: A Database of Protein Structure Alignments for Homologous Families. *Protein Science* vol. 7, 1998, pp. 2469–2471.

[18] S. B. Needleman, C. D. Wunsch. A General Method Applicable to the Search of Similarities in the Amino Acid Sequences of Two Proteins. *Journal of Molecular Biology* vol. 48, 1970, pp. 443–453.

[19] C. Notredame, D. G. Higgins. SAGA: Sequence Alignment by Genetic Algorithm. *Nucleic Acids Research* vol. 2, 1996, pp. 1515–1524.

[20] C. Notredame, D. G. Higgins, J. Heringa. T-COFFEE: A Novel Method for Fast and Accurate Multiple Sequence Alignment. *Journal of Molecular Biology* vol. 302, 2000, pp. 205–217.

[21] S. D. Prestwich. Negative Effects of Modeling Techniques on Search Performance. *Annals of Operations Research* vol. 18, Kluwer Academic Publishers, 2003, pp. 137–150.

[22] S. D. Prestwich. Randomised Backtracking for Linear Pseudo-Boolean Constraint Problems. *Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems*, le Croisic, France, 2002, pp. 7–20.

[23] S. D. Prestwich, D. Higgins, O. O'Sullivan. A SAT-Based Approach to Multiple Sequence Alignment. Poster, *Ninth International Conference on Principles and Practice of Constraint Programming*, Kinsale, Ireland, 2003 (to appear).

[24] K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, J. Kececioglu. A Branch-and-Cut Algorithm for Multiple Sequence Alignment. *First Annual International Conference on Computational Molecular Biology*, 1997, pp. 241–249.

[25] B. Selman, H. Kautz, B. Cohen. Noise Strategies for Improving Local Search. *Twelfth National Conference on Artificial Intelligence*, AAAI Press 1994, pp. 337–343.

[26] B. M. Smith, K. E. Petrie, I. P. Gent. Models and Symmetry Breaking for Peaceable Armies of Queens. *Workshop on Modelling and Solving Problems with Constraints, ECAI*, 2002.

[27] B. Smith, K. Stergiou, T. Walsh. Using Auxiliary Variables and Implied Constraints to Model Non-Binary Problems. *Seventeenth National Conference on Artificial Intelligence and Twelfth Innovative Applications of AI Conference*, 2000, pp. 182–187.

[28] J. D. Thompson, D. G. Higgins, T. J. Gibson. CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment Through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice. *Nucleic Acids Research* vol. 22, 1994, pp. 4673–80.

[29] J. P. Walser. Solving Linear Pseudo-Boolean Constraints With Local Search. *Eleventh Conference on Artificial Intelligence*, 1997, pp. 269–274.

[30] T. Walsh. SAT v CSP. *Sixth International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 1894, Springer-Verlag 2000, pp. 441–456.

```
1mnc  ------GPKWERTNLTYRIRNYTPQLSEAEVERAIKDAFELWSVASPLIFTGISQ-----
1hfc  -------PRWEQTHLTYRIENYTPDLPRADVDHAIEKAFQLWSNVTPLTFTKVSE-----
1bqoa YSLFPNSPKWTSKVVTYRIVSYTRDLPHITVDRLVSKALNMWGKEIPLHFRKVVW-----
456ca FRTFPGIPKWRKTHLTYRIVNYTPDLPKDAVDSAVEKALKVWEEVTPLTFSRLYE-----
1mmpa ------TLKWSKMNLTYRIVNYTPDMTHSEVEKAFKKAFKVWSDVTPLNFTRLHD-----
1bqqm ----IQGLKWQHNEITFCIQNYTPKVGEYATYEAIRKAFRVWESATPLRFREVPYAYIRE

1mnc  G---EADINIAFYQRDHGDGSPFDGPNGILAHAFQPGQGIGGDAHFDAEETWTNTSA---
1hfc  G---QADIMISFVRGDHRDNSPFDGPGGNLAHAFQPGPGIGGDAHFDEDERWTNNFR---
1bqoa G---TADIMIGFARGAHGDSYPFDGPGNTLAHAFAPGTGLGGDAHFDEDERWTDGSS-L-
456ca G---EADIMISFAVREHGDFYPFDGPGNVLAHAYAPGPGINGDAHFDDDEQWTKDTT---
1mmpa G---IADIMISFGIKEHGDFYPFDGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSK---
1bqqm GHEKQADIMIFFAEGFHGDSTPFDGEGGFLAHAYFPGPNIGGDTHFDSAEPWTVRNEDLN

1mnc  NYNLFLVAAHEFGHSLGLAHSSDPGALMYPNYAF-RETSNYSLPQDDIDGIQAIYG----
1hfc  EYNLHRVAAHELGHSLGLSHSTDIGALMYPSYTFSGD---VQLAQDDIDGIQAIYGRS--
1bqoa GINFLYAATHELGHSLGMGHSSDPNAVMYPTYGN-GDPQNFKLSQDDIKGIQKLYGK---
456ca GTNLFLVAAHEIGHSLGLFHSANTEALMYPLYHSLTDLTRFRLSQDDINGIQSLYGPPPD
1mmpa GYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTY-TGKSHFMLPDDDVQGIQSLYGPG--
1bqqm GNDIFLVAVHELGHALGLEHSSDPSAIMAPFYQW-MDTENFVLPDDDRRGIQQLYGGES-
```

Figure 2: Result for the mmp problem

```
1frfs ---KHRPSVVWLHNAECTGCTEAAIRTIKPYIDALILDTISLDYQETIMAAAGETSEAAL
2frva ---KKRPSVVYLHNAECTGCSESVLRTVDPYVDELILDVISMDYHETLMAGAGHAVEEAL
1e3da ----SRPSVVYLHAAECTGCSEALLRTYQPFIDTLILDTISLDYHETIMAAAGEAAEEAL
1h2rs LMGPRRPSVVYLHNAECTGCSESVLRAFEPYIDTLILDTLSLDYHETIMAAAGDAAEAAL
1cc1s ----KKAPVIWVQGQGCTGCSVSLLNAVHPRIKEILLDVISLEFHPTVMASEGEMALAHM

1frfs HEALEGKDG-YYLVVEGGLPTIDGGQWGMVAG-------HPMIETCKKAAAKAKGIICIG
2frva HEAIKGD---FVCVIEGGIPMGDGGYWGKVGG-------RNMYDICAEVAPKAKAVIAIG
1e3da QAAVNGPDG-FICLVEGAIPTGMDNKYGYIAG-------HTMYDICKNILPKAKAVVSIG
1h2rs EQAVNSPHG-FIAVVEGGIPTAANGIYGKVAN-------HTMLDICSRILPKAQAVIAYG
1cc1s YEIAEKFNGNFFLLVEGAIPTAKEGRYCIVGEAKAHHHEVTMMELIRDLAPKSLATVAVG

1frfs TCSPYGGVQKAKPNPSQAKGVSEAL---G--VKTINIPGCPPNPINFVGAVVHVLT----
2frva TCATYGGVQAAKPNPTGTVGVNEALGKLG--VKAINIAGCPPNPMNFVGTVVHLLT----
1e3da TCACYGGIQAAKPNPTAAKGINDCYADLG--VKAINVPGCPPNPLNMVGTLVAFLK----
1h2rs TCATFGGVQAAKPNPTGAKGVNDALKHLG--VKAINIAGCPPNPYNLVGTIVYYLKN---
1cc1s TCSAYGGIPAAEGNVTGSKSVRDFFADEKIEKLLVNVPGCPPHPDWMVGTLVAAWSHVLN

1frfs K---GIPDLDENGRPKLFYGELVHDNCPRLPHFEASEFAPSFDSEEAKKGFCLYELGCKG
2frva K---GMPELDKQGRPVMFFGETVHDNCPRLKHFEAGEFATSFGSPEAKKGYCLYELGCKG
1e3da G---QKIELDEVGRPVMFFGQSVHDLCERRKHFDAGEFAPSFNSEEARKGWCLYDVGCKG
1h2rs K---AAPELDSLNRPTMFFGQTVHEQCPRLPHFDAGEFAPSFESEEARKGWCLYELGCKG
1cc1s PTEHPLPELDDDGRPLLFFGDNIHENCPYLDKYDNSEFAETFTK-----PGCKAELGCKG

1frfs PVTYNNCPKVLFNQ-VNWPVQAGHPCLGCSEPDFWDTMTPFYEQG
2frva PDTYNNCPKQLFNQ-VNWPVQAGHPCIACSEPNFWDLYSPFYSA-
1e3da PETYNNCPKVLFNE-TNWPVAAGHPCIGCSEPNFWDDMTPFYQN-
1h2rs PVTMNNCPKIKFNQ-TNWPVDAGHPCIGCSEPDFWDAMTPFYQN-
1cc1s PSTYADCAKRRWNNGINWCVEN-AVCIGCVEPDFPDGKSPFYVAE
```

Figure 3: Result for the oxidored_q6 problem