Improving Multiple Sequence Alignments with Constraint Programming and Local Search

Marco Correia, Fábio Madeira, Pedro Barahona, and Ludwig Krippahl

CENTRIA-DI {mc,fmmm,pb,ludi}@di.fct.unl.pt

Abstract. Sequence alignment is a core problem in Bioinformatics, and multiple sequence alignments (MSA) are an important tool for phylogenetics, motif and domain identification, physiological studies and even protein structure and interaction, since MSA provide information on the coevolution of amino acid residues. However, the complexity of simultaneously aligning multiple sequences is exponential on the number of sequences, and so MSA must be computed using heuristics that cut through this large search space, compromising the quality of the result and limiting the scoring functions that can be used. In this paper, we propose a constraint programming (CP) and local search based method for repairing MSA obtained with classical algorithms in order to improve the alignments and to allow greater flexibility in the scoring functions.

1 Introduction

The sequence alignment problem is at the origin of bioinformatics [1, 2] and is still of central importance. As the growth of sequence databases made more demands on the organization of these data, alignment algorithms that speed up the search by pruning the search space with heuristics became the norm [3, 4], dominating the field over the initial, dynamic-programming, approaches. This is even more evident in multiple sequence alignment (MSA), which is the harder problem of aligning, simultaneously, a large number of sequences.

A MSA can be understood as a graph where the symbols are joined by edges representing the correspondences in the alignment, with a complete n-partite graph with edges joining all elements of each sequence to all elements of every other sequence representing the set of all possible MSA [5, 6]. Alternatively, a MSA can be seen as a matrix where each row represents a sequence and each column represents a set of aligned elements from all sequences. In this case, gaps are added to the alignment matrix to adjust the position of the elements (gaps are not part of the sequences themselves). The ultimate goal of the MSA is to show the evolutionary relations between the sequences, indicating which parts of the molecules descend from common ancestors, and which were added or deleted during evolution.

Since the classical dynamic programming methods are exponential in the number of sequences, the practical solution is to limit the search. The Clustal family of MSA algorithms [7], for example, aligns all pairs of sequences and

then builds the MSA by progressively aligning the closest sequences with the consensus sequence obtained from previous alignments. With no possibility of backtracking, this greedy optimization is likely to stop at local optima, resulting in some misalignment. Furthermore, the scoring function cannot consider the complete MSA because the MSA is being built progressively. Some alternatives (e.g. MUSCLE [8], MAFFT [9] and PROBCONS [10]) can redo previous alignments, repeatedly divide an alignment into two groups of aligned sequences and then realigning the groups. Aside from these progressive methods in the Clustal family (including T-Coffee[11], which uses a similar approach), there are alternatives based on probabilistic models (e.g. hidden Markov models (HMM) [12]) and genetic algorithms (e.g. SAGA [13]). HMM methods are based on a variant of the character frequency profile matrices, taking into account position-specific insertion and deletion (indel) probabilities. GA methods stochastically combine and mutate candidate alignments through a directed evolutionary process by providing a measure of fitness for individual alignments within the population, but this is generally too slow for real applications [13].

However, in all cases there is the need to avoid most of the search space, compromising the quality of the final results. Also, the most used approaches are restrictive in the scoring functions that can be used. For instance, one problem of particular interest to us is the detection of protein coevolution by the correlation of mutations in different positions. A MSA can show these correlations, indicating that those residues coevolved because of an important interaction. This can have structural implications and is useful in predicting protein structure and interaction. However, typically the MSA alignment scores assume that all mutations are independent, an assumption that is false whenever there is coevolution, and, in general, the MSA algorithms depend on such assumptions. For example, it would be infeasible to simply add this score to a progressive algorithm like ClustalW, which assembles the MSA one sequence at a time, since we need the alignment to estimate the correlation.

Evidence for the need to improve the MSA generated automatically can be found in the manual fixing of misalignments ("by eye") often reported in the literature, where researchers adjust the MSA according to their own criteria. Manually refined alignments are generally considered superior to purely automated methods [14], taking into account structural and functional factors. The assessment of MSA generally include the effectiveness of a particular heuristic for the optimization of the scoring function and the accuracy relative to reference alignments [15]. One of the best databases of manually refined and curated MSA, specifically designed for the evaluation and comparison of MSA software, is BAliBASE [16]. BAliBASE provides manually refined alignments based on 3D structural superpositions and implements two different alignment scores. The sum-of-pairs (SP) score which is the percentage of correctly aligned pairs of residues in the test alignment, relative to the reference alignment, to determine the success in aligning some, if not all, of the sequences in an alignment. And the column score (CS), the percentage of correctly aligned columns, which tests

the ability of the programs to align all of the sequences correctly at any given position.

1.1 Our proposal

Some methods have been proposed to solve alignment problems using constraint programming (CP) or related approaches, such as a CP problem in order to introduce additional constraints in sequence alignments [17], as an integer linear programming problem [18] or as a SAT problem [19]. Our proposal differs in that we take advantage of the established methods, such as ClustalW, and then repair the MSA using a CP approach, focusing on those regions that, being less conserved, are more likely to include misalignments. In this way we can narrow down the search space based on constraints such as not lowering the alignment score, we can impose additional constraints, such as those based on structure comparisons, and we can use a greater range of scoring functions than those available to progressive methods. Furthermore, this approach is closer to the established practice in the biological community of obtaining the MSA using the automated methods and then refining it according to the additional considerations. The main contribution of this paper is this framework for improving the MSA by undoing mistakes made by the greedy heuristics and allowing other scoring functions that may need to consider the MSA as a whole, such as mutual information across different positions.

2 Method

The problem of correcting a region of a MSA may be formalized as a matrix of $n \times p$ cells representing amino acid codes, where n is the number of sequences to be aligned and p are possible positions for the amino acid residues. Gaps in the sequence are represented by the special character "-" (fig.1).

S	P	V	I	-	-	-	L
R	-	-	I	-	-	-	S
S	-	-	-	-	-	-	L
F	N	T	T	Q	G	G	P
T	-	-	-	-	-	-	-
F	S	K	N	-	-	-	-
E	T	F	G	Q	-	-	-
K	S	-	-	-	-	-	T

Table 1. Multiple sequence alignment problem

Let $a_{i,j}$ represent the residue (or gap) at sequence i, position j, and $\mathbf{s}_i$ represent the sequence of amino acid residues at row i, i.e. $\mathbf{s}_i = \langle a_{i,1}, \ldots, a_{i,p} \rangle$. Let the score associated with two residues $a_{1,j}$,$a_{2,j}$ in the same position j given by a

function $\sigma_A(a_{1,j}, a_{2,j})$. From the individual amino acid scores we may compute a score for the alignment of two sequences $\mathbf{s}_1$, $\mathbf{s}_2$,

$$\sigma_S(\mathbf{s}_1, \mathbf{s}_2) = \sum_{i=1}^{p} \sigma_A(a_{1,i}, a_{2,i}) - \gamma(\mathbf{s}_1, \mathbf{s}_2) \tag{1}$$

The term $\gamma(\mathbf{s}_1, \mathbf{s}_2)$ present in the formula above is called *gap penalty* and accounts for the number of consecutive gaps in the sequences. The score of the multiple alignment is then derived from the pairwise sequence alignment scores,

$$\sigma = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sigma_S(\mathbf{s}_i, \mathbf{s}_j)$$

The alignment correction problem is to find the alignment with the best score by changing the positions of the gaps. Note that this procedure may increase or decrease the number of consecutive gaps (for example the number of consecutive gaps in the second sequence in fig.1 may be decreased from 2 to 1 by placing "I" next to "R" or before "S").

2.1 CP Model

The alignment correction problem may be naturally modeled in Constraint Programming. A straightforward approach assigns a finite domain variable $x_{i,j} \in X$ for each cell in the matrix, where its domain $D(x_{i,j}) = \{'-', 'A', 'C', 'D', \ldots\}$ is the set of all possible amino acids plus the gap. Each sequence $\mathbf{s}_i$ may be obtained by changing the position of the gaps. This constraint, which we call validSequence, may be modeled by the inTable constraint [20] as follows,

$$\text{validSequence}(\mathbf{s}_i) = \text{inTable}(\langle x_{i,1}, \ldots, x_{i,p} \rangle, T_i) \tag{2}$$

Each table T_i is created so that each row is obtained by placing the gaps in a distinct position. The number of rows in table T_i is therefore $C_{g_i}^{p}$ where g_i is the number of gaps in sequence $\mathbf{s}_i$. Since the complexity of the propagation algorithm for this constraint is linear on the size of the table this method works only for small p. Fortunately, we may propagate this constraint using an algorithm that is not exponential in p as follows.

For each sequence $\mathbf{s}_i$ we introduce a set X_i^G of auxiliary finite domain variables, where a variable $x_{i,k}^G \in X_i^G$ models the position of the k'th gap in the sequence $\mathbf{s}_i$, and hence $1 \leq k \leq g_i$. Similarly, we introduce a second set X_i^A of auxiliary finite domain variables, where a variable $x_{i,k}^A \in X_i^A$ represents the position of the k'th amino acid residue in the sequence $\mathbf{s}_i$, which we denote as $\mathbf{s}_i(k)$, with $1 \leq k \leq p - g_i$. Note that these two sets are strictly sorted, and partition the set $\{1, \ldots, p\}$. The validSequence constraint is therefore modeled as follows,

$$\begin{aligned}\textsc{validSequence}\left(\mathbf{s}_i\right) = & \forall_{2\leq k\leq g_i} x^G_{k-1} < x^G_k \\ & \wedge \forall_{2\leq k\leq p-g_i} x^A_{k-1} < x^A_k \\ & \wedge \textsc{distinct}\left(X^G_i \cup X^A_i\right) \\ & \wedge \forall_{1\leq k\leq g} X_i\left[x^G_{i,k}\right] =' -' \\ & \wedge \forall_{1\leq k\leq <p-g} X_i\left[x^A_{i,k}\right] = \mathbf{s}_i\left(k\right) \end{aligned} \tag{3}$$

For modeling the objective function we use formula 1, and apply function σ_A over all pairs of finite domain variables $x_{1,j}$, $x_{2,j}$ in the same column. This may be accomplished by means of a TABLE or ELEMENT constraint over the table of amino acid affinities. Finally, the gap penalty term $\gamma\left(\mathbf{s}_1, \mathbf{s}_2\right)$ may be integrated also using different models. If the VALIDSEQUENCE constraint is implemented by equation 2, then we can create an extra column in each row of a table T_i to specify the number of consecutive gaps corresponding to the given sequence, and project this number to a new finite domain variable c_i by changing equation 2 to

$$\textsc{validSequence}(\mathbf{s}_i)=\textsc{inTable}\left(\left\langle x_{i,1}, \ldots, x_{i,p}, c_i\right\rangle, T_i\right)$$

When using equation 3 to model the VALIDSEQUENCE constraint, c_i may be obtained using reification and a SUM constraint as follows,

$$c_i = \sum_{k=2}^{g_i} \left[x^G_{i,k} > x^G_{i,k-1} + 1\right]$$

2.2 Search

We used a greedy variable and value heuristics for directing search quickly towards a good solution. They are defined as

$$\begin{aligned}\textsc{var}\left(X\right) &= \arg\max_{x_{i,j}} \max_{v_1, v_2 \in D(x_{i,j})} q\left(x_{i,j}, v_1\right) - q\left(x_{i,j}, v_2\right) \\ \textsc{val}(x_{i,j}) &= \arg\max_{v\in D(x_{i,j})} q\left(x_{i,j}, v\right)\end{aligned}$$

where $q\left(x_{i,j}, v\right)$ is a function which estimates the cost of assigning value v to variable $x_{i,j}$,

$$q\left(x_{i,j}, v\right) = q^A\left(x_{i,j}, v\right) + nq^G\left(x_{i,j}, v\right) \tag{4}$$

and is composed of two terms. The first estimates the cost of this assignment based on the set of amino acid residues already placed in column j, using function σ_A,

$$q^A\left(x_{i,j}, v\right) = \sum_{k=1}^{n} \begin{cases} \sigma_A\left(x_{k,j}, v\right) & \Leftarrow \left|D\left(x_{k,j}\right)\right| = 1 \\ 0 & \Leftarrow \text{otherwise} \end{cases}$$

The second term estimates the impact that this assignment will have on the number of consecutive gaps and is -10 if it creates a new gap, 10 if it does not open a new gap, and 0 if it is not known.

The above heuristics were used to drive limited discrepancy search [21]. The allowed discrepancy begins at 1 and is iteratively increased each time the search space is exhausted so that completeness is still guaranteed. Additionally, since the search space appears to have some steep local optima, we introduced a small random perturbation term in formula 4, and restarted the solver after a geometrically increasing slice of time.

2.3 Local Search

The same model was tested in constrained local search (COMET), with a greedy hill-climbing heuristic optimizing the same score using the Gonnet substitution matrix (the objective function) starting from a randomized transformation of the MSA blocks. With a few exceptions the improvements on the score were not as good as those obtained with the CP model and heuristics, but (like for CP) better heuristics and meta-heuristics should improve the obtained scores. These results should thus be regarded as initial, and subject to further work.

2.4 Experiments

To test our approach, we measured how much we could improve the standard ClustalW MSA, both according to a scoring function and by comparing the alignments to the reference alignments in BALiBASE. The procedure was thus to start from the same set of sequences as those in a BALiBASE reference alignment, obtain the ClustalW MSA and attempt to improve the alignment on contiguous blocks of columns containing a mix of gaps and residues. This ruled out regions that are well conserved and thus more likely to be correctly aligned, and allowed us to focus on the regions of the MSA that could be adjusted by shifting the gap positions. Our scoring function to improve the alignments was the standard Gonnet substitution matrix [22], and the ClustalW alignments were calculated with this matrix and the default gap penalties. The next step was to apply the CP and Local Search algorithms to improving the alignment score. However, one problem is that ClustalW uses a highly optimized scoring function that, though based on the same substitution matrix we used, adjusts the relative weights given to each sequence and also the gap penalties depending on the neighboring residues [23]. This means that our improvement could come either from actually correcting errors in the ClustalW alignment or due simply to the slight differences in the scoring function.

To test this, we compared the ClustalW and corrected alignments with the BALiBASE set of highly accurate alignments, manually curated by experts and determined not only from sequence data but also from structural information. Given that ClustalW uses a more sophisticated scoring function than our current implementation, and since neither ClustalW nor our implementation is using structural data or other information available to the experts that created the BALiBASE alignments, if it were the case that our score improvements were only due to a difference in the scoring functions we would expect our corrections not

to improve the ClustalW alignments when compared to the BALiBASE alignments. In contrast, if, even with a simpler scoring function, our implementation was improving the alignments by making them more similar to the BALiBASE benchmark, this would mean that our approach was really correcting mistakes made by ClustalW. We applied this procedure to 22 alignments for the CP implementation and 24 for the local search implementation (a few were rejected because they provided less than 10 columns for adjustment over the whole alignment). Running times ranged from a few seconds per alignment in the local search implementation to several minutes for the CP implementation, which was set to time out at two minutes for each block.

3 Results and Discussion

For the CP implementation, with the Gonnet substitution matrix score the average improvement for each column changed in the alignments (up to 233 columns, in one case, but averaging 54 columns for the set of 22 MSAs) was slightly above the average score attributed to the match of identical residues (106% of the average identity match score in the substitution table). This means that our improvement was the equivalent of gaining one additional identity match for each column changed. In the comparison with the reference alignments using the BALiBASE sum-of-pairs score, our implementation improved 77% (17 out of 22) of the alignments tested tests using CP. This suggests that, even with a simpler scoring function and no additional information such as structural or functional data, the CP implementation improves the ClustalW alignment. The average improvement for all 22 alignments was 11% in the BALiBASE sum-of-pairs score. The local search implementation did not perform as well, with an insignificant average improvement when compared to BALiBASE alignments and improving only 14 out of 24 alignments. Nevertheless, our goal in this paper was to show that improving MSA by correcting mistakes in less conserved regions is a promising approach. At this stage CP seems to give better results, but there is still work to be done optimizing the scoring functions and heuristics, and the local search implementation takes only milliseconds to find solutions, against several minutes for CP, so there is much room for improvement. In addition, this gives us a flexible framework for using different scoring functions, not limited to the peculiarities of the underlying alignment algorithm. Of special interest is the inclusion of structural information, whether from determined structures or prediction algorithms, and also scoring functions adapted to coevolution studies, where the assumption of independent mutations does not hold.

Acknowledgements

This work was funded by Fundação para a Ciência e Tecnologia, MCTES, under project PTDC/EIA-CCO/115999/2009.

References

[1] Needleman SB and Wunsch CD. A general method applicable to the search for similarities in [...] proteins. J. Mol. Biology 48 (3): 443–53 (1970)

[2] Smith TF, Waterman MS.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)

[3] Lipman, DJ; Pearson, WR. Rapid and sensitive protein similarity searches. Science 227 (4693): 1435–41 (1985)

[4] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol 215 (3): 403–410 (1990)

[5] Kececioglu, JD. The maximum weight trace problem in multiple sequence alignment. In Apostolico et al (eds.), Proc. 4th Symp. Comb. Patt. Matching, pp 106–119 (1993).

[6] Backofen R., Gilbert D., Bioinformatics and Constraints. In: Rossi F., van Beek, P., Walsh T. (eds.) Handbook of Constraint Programming, Elsevier 2006.

[7] Chenna R, Sugawara H, Koike T et al. Multiple sequence alignment with the Clustal series of programs. Nucleic Acids Res 31 (13): 3497–3500 (2003)

[8] Edgar, R.C. MUSCLE: multiple sequence alignment with high accuracy and high throughput.Nucleic Acids Res. 32(5):1792-1797 (2004)

[9] Katoh K, Toh H. Recent developments in the MAFFT multiple sequence alignment program. Brief Bioinform 9: 286–298 (2008)

[10] Do CB, Mahabhashyam MS, Brudno M, Batzoglou S. ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Res 15: 330–340 (2005)

[11] Notredame C, Higgins DG, Heringa J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. J. of mol. biol., 302(1), 205-17. (2000)

[12] Eddy, S. R. Profile hidden Markov models. Bioinformatics, 14(9), 755. Oxford Univ Press (1998)

[13] SAGA: Sequence Alignment by Genetic Algorithm, C. Notredame, D.G. Higgins, Nucleic Acid Research, Vol. 24, 1515-1524, (1996)

[14] Edgar, R. C., Batzoglou, S. Multiple sequence alignment. Current opinion in structural biology, 16(3), 368-73 (2006)

[15] Do, CB, Katoh K, Protein Multiple Sequence Alignment, Functional Proteomics Methods in Molecular Biology, 2008, Volume 484, IV, 379-413

[16] Thompson, J. D., Plewniak, F., Poch, O. BAliBASE: a benchmark alignment database [...]. Bioinformatics (Oxford, England), 15(1), 87-8 (1999)

[17] Will S., Bush A., Backofen R. Efficient Sequence Alignment with Side-Constraints by Cluster Tree Elimination. Constraints 13(1-2): 110-129 (2008)

[18] Reinert K, Lenhof HP, Mutzel P, Mehlhorn K, Kececioglu JD, A Branch-and-Cut Algorithm for Multiple Sequence Alignment, In Proc. of the 1st RECOMB (1997)

[19] Prestwich S, Higgins D. A SAT-Based Approach to Multiple Sequence Alignment. In 9th Int. Conf. Princ. and Pract. of CP, pp. 940–944 (2003)

[20] Bessière C and Régin JC. Arc Consistency for General Constraint Networks: Preliminary Results. IJCAI'97 pp 398—404 (1997)

[21] Harvey WD, Matthew LG. Limited Discrepancy Search, In C. S. Mellish (ed) Proceedings of IJCAI'95; Vol. 1, pages 607–615 (1995)

[22] Gonnet GH, Cohen MA, Benner SA. Exhaustive matching of the entire protein sequence database. Science, 256(5062):1443-5. (1992)

[23] Thompson JD, Higgins DG, Gibson TJ. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment [...]. Nucleic acids research, Vol. 22, No. 22, pp. 4673-4680. (1994)