Efficient Constraint-based Sequence Alignment by Cluster Tree Elimination*

Sebastian Will, Anke Busch and Rolf Backofen

Bioinformatics, Institute of Computer Science, Friedrich-Schiller-University
Ernst-Abbe-Platz 2, 07743 Jena, {*will,busch,backofen*}*@inf.uni-jena.de*

Abstract. Aligning DNA and protein sequences has become a standard method in molecular biology. Often, it is desirable to include partial prior knowledge and conditions in an alignment. The most common and successful technique for efficient alignment algorithms is dynamic programming (DP). However, a weakness of DP is that one cannot include additional constraints without specifically tailoring a new DP algorithm. Here, we discuss a declarative approach that is based on constraint techniques and show how it can be extended by formulating additional knowledge as constraints. We take special care to obtain the efficiency of DP for sequence alignment. This is achieved by careful modeling and applying proper solving strategies.

1 Introduction

Modern molecular biology is not possible without tools for the comparison of the macromolecules DNA, RNA, and proteins. It is most desirable to be able to specify additional restrictions for such similarity search whenever prior knowledge on the analyzed molecules is available. For example, consider the case of a biologist, who knows that certain regions in her sequences share a common local motif. Based on this knowledge, the rest of the sequences should be compared. Then, we need to optimize similarity under the additional constraint that parts of such regions should be matched to each other. Another striking example is the enhancement of RNA or protein comparison by employing knowledge on the structure of the macromolecules [10,3,1,5].

However in general, similarity searching tools on the web do not allow to take such prior knowledge into account automatically. The reason for this deficiency is of algorithmic nature. Only for certain special constraints, alignment algorithms have been discussed. In particular, there are approaches that incorporate anchor constraints [7] and precedence constraints [8]. We will later discuss how such constraints fit into our newly introduced framework as simple cases. Aligning sequences and (to some extent) sequences with additional structural information is commonly and most successfully performed by *dynamic programming (DP)* [9,11,3]. There is no straightforward and general way to extend a DP algorithm in order to take additional knowledge into account.

*This work is partially supported by the EU Network of Excellence REWERSE.

To overcome this, declarative formulations of the alignment problem have been proposed. Due to their use of constraints, such approaches can be extended to incorporate prior knowledge. For this aim, such knowledge is formulated as constraints and added to the model for unconstrained alignment. One such previous approach [6] is based on *integer linear programming (ILP)*. Since in ILP one can only use boolean variables, the ILP model of [6] for aligning two sequences of length n and m introduces $O(nm)$ variables for modeling the alignment edges. Due to the resulting complexity, one needs to introduce artificial restrictions on the possible alignment edges for solving the problem in practice. Furthermore, the solving strategy for ILP does not achieve the efficiency of DP for the unconstrained case. Another declarative approach [12] is based on constraint programming. The approach introduces quadratically many variables and constraints and remodels the given DP algorithm. As a consequence, only a rather restricted class of side constraints can be handled efficiently.

Here, we introduce a new constraint-based approach. The main challenge that we face with our approach is to compete with the very good efficiency of DP in the standard case and allow extension by introducing new constraints. We achieve the desired efficiency and adaption to additional constraints by modeling the alignment problem as a constraint optimization problem in the sense of [2,4] and then applying a special solution strategy, which is known as *cluster tree elimination (CTE)* [4].

2 A Constraint Model for Sequence Alignment

We develop a constraint model for sequence alignment of two sequences $a = a_1 \dots a_n$ and $b = b_1 \dots b_m$ that are both words of the alphabet Σ. To be more precise, we define an *alignment $\mathcal{A}$ of a and b* as an ordered matching of positions in a and b, i.e. as a subset of $\{1, \dots, n\} \times \{1, \dots, m\}$ such that for all $(i, j), (i', j') \in \mathcal{A}$:

1. $i = i'$ if and only if $j = j'$ and
2. $i < i'$ implies $j < j'$.

We call i and j matched by $\mathcal{A}$ if and only if $(i, j) \in \mathcal{A}$.

The *score of an alignment* $\mathcal{A}$, which we want to maximize, depends on the *similarity function on positions* $\sigma : \{1, \dots, n\} \times \{1, \dots, m\} \to \mathbb{R}$ and *gap cost* $\gamma \in \mathbb{R}$. It is defined as

$$\text{score}(\mathcal{A}) = (n + m - 2|\mathcal{A}|)\gamma + \sum_{(i,j)\in\mathcal{A}} \sigma(i, j). \tag{1}$$

The classical DP algorithm for sequence alignment is specified via the recursion equation $D_{i\,j} = \max\{D_{i-1\,j-1} + \sigma(i, j), D_{i-1\,j} + \gamma, D_{i\,j-1} + \gamma\}$ with initialization $D_{0\,0} = 0$, $D_{i\,0} = i\gamma$, and $D_{0\,j} = j\gamma$ for $1 \le i \le n$ and $1 \le j \le m$ and solves the problem in $O(nm)$ time.

Here, we model alignment as a constraint optimization problem in the framework that is described in a more general form in [4]. There, one defines variables

with finite domains and functions on these variables. In our special case, the solution of the problem is a valuation of the variables that maximizes the sum of the function values. Note that hard constraints c can be encoded in this framework by functions that yield $-\infty$ if the constraint is violated and 0 otherwise. Tacitly, our arithmetic is extended canonically in order to handle sums and maximizations involving infinity.

In our model, we represent alignments of a and b by finite domain variables X_i for $1 \leq i \leq n$ with domains $\mathrm{dom}(X_i) = \{0, \ldots, m\}$. Furthermore for technical reasons, we introduce the fixed variables $X_0 = 0$ and $X_{n+1} = m + 1$ and extend σ by defining $\sigma(n + 1, m + 1) = 0$. A given alignment $\mathcal{A}$ is uniquely encoded by a valuation $(X_0 = x_0, \ldots, X_{n+1} = x_{n+1})$ of variables $X_0, \ldots, X_{n+1}$ where 1.) $x_i = j$ if $(i, j) \in \mathcal{A}$ and 2.) $x_i = x_{i-1}$, for every i that is not matched in $\mathcal{A}$. Note that i and j are matched if and only if $x_i = j$ and $x_i > x_{i-1}$. For example, the valuation $\boldsymbol{x} = (0, 1, 2, 5, 6, 6, 7, 8)$ of $X_0, \ldots, X_8$ corresponds to the alignment $\{(1,1),(2,2),(3,5),(4,6),(7,7)\}$, which can be represented alternatively by

$$\begin{array}{c} a_1\, a_2\, -\, -\, a_3\, a_4\, a_5\, a_6\, a_7 \\ b_1\, b_2\, b_3\, b_4\, b_5\, b_6\, -\, -\, b_7 \end{array}.$$

The only hard constraints on the variables X_i are $X_{i-1} \leq X_i$ for $1 \leq i \leq n+1$. They are modeled by functions

$$leq_i : \mathrm{dom}(X_{i-1}) \times \mathrm{dom}(X_i) \to \{-\infty, 0\}.$$

The scoring scheme is encoded via functions $f_i(X_{i-1}, X_i)$ for $1 \leq i \leq n + 1$ that are defined by

$$f_i(j', j) = \begin{cases} \sigma(i, j) + (j - j' - 1)\gamma & \text{if } j' < j \\ \gamma & \text{otherwise.} \end{cases}$$

Note that we correctly model alignments and their scores. Firstly, a valuation $(X_0 = x_0, \ldots, X_{n+1} = x_{n+1})$ represents an alignment $\mathcal{A}$ of a and b if and only if $\sum_{1 \leq i \leq n+1} f_i(x_{i-1}, x_i) + leq_i(x_{i-1}, x_i)$ is not $-\infty$. Secondly in this case, $\sum_{1 \leq i \leq n+1} f_i(x_{i-1}, x_i)$ equals the score of $\mathcal{A}$ (see Eq. 1).

3 Efficient Solving by Cluster Tree Elimination

Here, we sketch CTE and show its application to the our model. We demonstrate how direct application of CTE yields an $O(nm^2)$ algorithm. Then, by introducing modifications to the standard CTE approach, we improve the complexity to $O(nm)$ time.

For applying CTE, we first need a *cluster tree decomposition (CTD)* [4]. In such a decomposition, we distribute variables and functions to vertices (*clusters*) of a tree, such that 1.) each function occurs in exactly one cluster, 2.) if a function occurs in a cluster, then all variables of the function are assigned to the cluster as well, and 3.) for each variable the set of clusters that contain this variable induces a connected subtree.

Due to the definition, clusters that share variables are connected by edges. The shared variables are called *separator variables*. Figure 1 shows a cluster tree decomposition of our alignment model where edges are labeled by separator variables. We call the cluster consisting of X_{i-1}, X_i, f_i, and leq_i the cluster i. Note that in this figure (and the following ones) we omit the functions leq_i in our presentation.

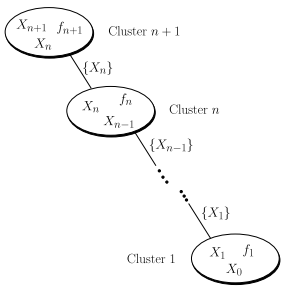

Fig. 1. CTD of pure sequence alignment.

CTE solves a constraint optimization problem by repeatedly exchanging messages between the clusters. The messages are functions that combine the functions of the cluster and marginalize them to the separator variables. Each message becomes a new function of the receiving cluster. From cluster i to cluster $i+1$, CTE sends a function g_i of the separator variable X_i. Beginning with cluster 1 it proceeds until cluster $n+1$ receives its message g_n. When sending a message from cluster i, this cluster is already augmented by a function g_{i-1}. Finally, it can be shown that $\max_{1\leq j\leq m}(g_n(j) + f_{n+1}(j, m+1))$, which is the marginalization of the functions in cluster $n+1$ to the empty set of variables, is the maximal alignment score.

It remains to show how the messages g_i are computed. Due to the CTE algorithm, the message g_i is defined for $0 \leq j \leq m$ as

$$g_i(j) = \max_{0\leq j'\leq m} \left(g_{i-1}(j') + f_i(j', j) + leq_i(j', j)\right). \tag{2}$$

Clearly, the standard approach takes $O(m^2)$ time for computing the function g_i. Since $O(n)$ messages are sent until the final alignment score can be computed, this results in an $O(nm^2)$ algorithm. Thereby, we have shown that the direct application of CTE to our constraint model yields a polynomial algorithm for sequence alignment.

Improving complexity. The complexity can be improved further if we employ the internal structure of the functions g_{i-1}, f_i, and leq_i. For this reason, we rewrite Equation 2 by the semantics of leq_i and expand the definition of f_i.

$$g_i(j) = \max_{0\leq j'\leq j} \left(g_{i-1}(j') + \begin{cases} \sigma(i,j) + (j - j' - 1)\gamma & \text{if } j' < j \\ \gamma & \text{otherwise} \end{cases}\right).$$

Now, we can resolve the case distinction of f_i and move the constant $\sigma(i, j)$ out of the maximization. Then,

$$g_i(j) = \max \begin{cases} \sigma(i,j) + \max\limits_{0\leq j'<j} \left(g_{i-1}(j') + (j - j' - 1)\gamma\right) \\ g_{i-1}(j) + \gamma. \end{cases}$$

A helper function $g^{\mathrm{m}}(j) = \max_{0 \le j' < j} (g_{i-1}(j') + (j - j' - 1)\gamma)$ can be defined recursively and then computed in $O(m)$ time by DP as

$$g^{\mathrm{m}}(0) = -\infty,\ g^{\mathrm{m}}(1) = g_{i-1}(0),\ \text{and for } j > 1\ g^{\mathrm{m}}(j) = \max \begin{cases} g^{\mathrm{m}}(j-1) + \gamma \\ g_{i-1}(j-1). \end{cases}$$

In consequence, the total computation of g_i is done in $O(m)$. This results in an $O(nm)$ time algorithm for the computation of the alignment score.[1]

4 Extension of the Sequence Alignment Model

Recently discussed constrained alignment approaches handled constraints like anchor constraints and precedence constraints. Such constraints can be encoded in our model straightforwardly and are handled by restricting the domains of variables, which even increases the efficiency of our algorithm. An *anchor constraints*, as discussed in [7], tells that position i in the first sequence can only be aligned to position j in the second sequence and furthermore, positions strictly left (resp. right) of i are aligned to positions strictly left (resp. right) of j. These conditions are expressed in our model by the constraints

$$X_{i-1} < j, X_{i+1} > j, \text{ and}$$
$$X_i = j \vee X_i = X_{i-1};$$

the latter implies $X_i \le j$. The constraints are directly propagated to the domains of X_i and X_{i-1} and do not increase the complexity of our constraint problem. Via the less than constraints the new domain information is further propagated to the domains of all variables.

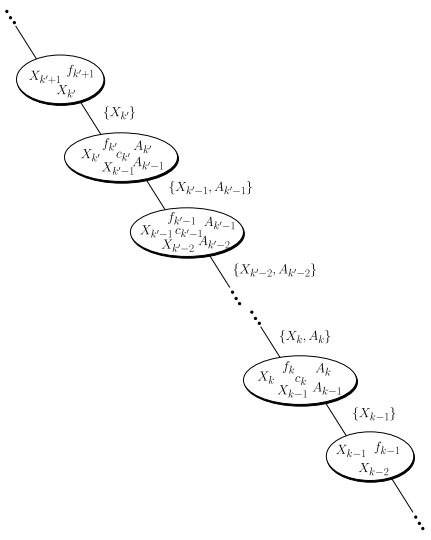

Fig. 2. CTD of an alignment with segment constraints.

A *precedence constraint*, handled in [8], tells that in the alignment position i of the first sequence is left (resp. right) of position j of the second sequence. In our model, corresponding conditions are encoded as $X_i \ge j$ (resp. $X_i \le j$). A further example for a trivial extension of the model is a condition like "position k is aligned to l or l'" (constraints: $X_k \in \{l, l'\}$ and $X_{k-1} < X_k$).

In this section, we discuss two more challenging extensions by example. Namely, the incorporation of prior knowledge on aligned segments and the extension to sequence structure alignment.

[1] $O(m)$ space can be achieved by further modifications to CTE.

Aligned Segments As example of constraining the alignment between segments in a and b, we consider the constraint that at least x% of the positions $\{k, \ldots, k'\}$ in a have to be matched with positions $\{l, \ldots, l'\}$ in b. For extending our model by this constraint, we add variables $A_{k-1}, \ldots, A_{k'}$ and for each $k \leq i \leq k'$ the function $c_i(A_{i-1}, A_i, X_{i-1}, X_i)$ that encodes the hard constraint

$$A_i = A_{i-1} + \begin{cases} 1 & \text{if } X_{i-1} < X_i \text{ and } l \leq X_i \leq l' \\ 0 & \text{otherwise.} \end{cases}$$

We fix $A_{k-1} = 0$. Since the variables A_i count the proper matches in the prefix segment {k,. . . ,i}, we can finally express the constraint by restricting the domains of A_i to $\{\max(0, \lceil \frac{x}{100}(k' - k + 1) \rceil - (k' - i)), \ldots, i - k + 1\}$.

Figure 2 shows the cut-out of the CTD that is affected by the extension of the model. CTE works essentially as in the standard case. For $k \leq i \leq k'$, CTE sends messages g_i depending on the separator variables that each can be computed in $O(m\bar{k})$ time where $\bar{k} = k' - k + 1$. Thus, the total complexity is $O(m(n - \bar{k}) + m\bar{k}^2)$. Note that, as assumed in this result, one can transfer the complexity improvement of the previous section to this case of constrained alignment. It suffices to look at the message g_i from the cluster that contains a variable A_i (and thus contains the variables X_i, X_{i-1}, and A_{i-1} by construction). The message g_i, which depends on values for X_i and A_i, is given (already using the semantic of leq_i and c_i) as

$$g_i(j, a) = \max_{0 \leq j' \leq j} \begin{cases} g_{i-1}(j', a - 1) + f_i(j', j) & \text{if } j' < j \text{ and } l \leq j \leq l' \\ g_{i-1}(j', a) + f_i(j', j) & \text{otherwise.} \end{cases}$$

One transforms further to

$$g_i(j, a) = \max \begin{cases} \sigma(i, j) + g^{\mathrm{m}}(j, a) \\ \gamma + g_{i-1}(j) \end{cases}$$

where we define

$$g^{\mathrm{m}}(j, a) = \max_{0 \leq j' < j} ((j - j' - 1)\gamma + g_{i-1}(j', a'))$$

where $a' = a$ if $l \leq j \leq l'$ and $a' = a - 1$ otherwise. Finally, g^{m} can be defined recursively as in the previous section as

$$g^{\mathrm{m}}(0, a) = -\infty, \; g^{\mathrm{m}}(1, a) = g(1, a'), \text{ and}$$
$$\text{for } j > 1 \; g^{\mathrm{m}}(j, a) = \max \begin{cases} g^{\mathrm{m}}(j - 1, a'') + \gamma \\ g_{i-1}(j - 1, a'') \end{cases}$$

where $a' = a$ if $l \leq 1 \leq l'$ and $a' = a - 1$ otherwise. Furthermore, $a'' = a$ if $l \leq j \leq l'$ and $a'' = a - 1$ otherwise.

We have demonstrated, that for this class of constraints the efficiency can be improved in the same way as in the case of unconstrained alignment. Intuitively, the additional constraints do not interfere with the nature of our score that enables the recursive decomposition.

Sequence Structure Alignment Here as additional input, we have two structures $P_a \subset \{1,\ldots,n\} \times \{1,\ldots,n\}$ and $P_b \subset \{1,\ldots,m\} \times \{1,\ldots,m\}$ and a function $\omega : \{1,\ldots,n\} \times \{1,\ldots,n\} \times \{1,\ldots,m\} \times \{1,\ldots,m\} \to \mathbb{R}$. A pair $(i_l, i_r) \in P_a$ (resp. $(j_l, j_r) \in P_b$) expresses a dependency, e.g. base pairing in RNA, between the positions i_l and i_r (resp. j_l and j_r). The function ω yields a score for aligning pairs of dependent positions.

The score of an alignment $\mathcal{A}$ is now defined in extension of Eq. 1 as

$$\text{score}(\mathcal{A}) + \sum_{\substack{(i_l,i_r)\in P_a,(j_l,j_r)\in P_b,\\ (i_l,j_l)\in\mathcal{A},(i_r,j_r)\in\mathcal{A}}} w(i_l, i_r; j_l, j_r).$$

Our alignment model can be extended by adding for each $(i_l, i_r) \in P_a$ functions $h_{i_l i_r}(X_{i_l-1}, X_{i_l}, X_{i_r-1}, X_{i_r})$ that are defined as

$$h_{i_l i_r}(j'_l, j_l, j'_r, j_r) = \begin{cases} \omega(i_l, i_r; j_l, j_r) & \text{if } j'_l < j_l,\ j'_r < j_r, \text{ and } (j_l, j_r) \in P_b \\ 0 & \text{otherwise.} \end{cases}$$

Figure 3 provides an example for $P_a = \{(k_l, k_r), (l_l, l_r)\}$ and arbitrary P_b, which demonstrates the general construction principle of such a CTD. Due to the base pair (k_l, k_r) (and analogously for (l_l, l_r)), the decomposition contains a node consisting of the variables X_{k_l}, X_{k_r} and their predecessors X_{k_l-1}, X_{k_r-1}, since these variables depend on each other via the function $h_{k_l k_r}$. This node is parent of two sub-trees. In its left sub-tree, we handle the alignment for positions between k_l and k_r and in the right sub-tree the alignment for the positions less than k_l. Due to the conditions for a CTD, the variable X_{k_l} has to be shared with nodes of the left sub-tree, since it is constrained to variables in the leftmost leave.

In this tree structure, CTE begins with the leave vertices and proceeds to the root. From each cluster, it sends a message to its parent cluster. The final alignment score is obtained from the root node.

5 Conclusion

We present the first declarative approach to sequence alignment that is equally efficient as the commonly used method of dynamic programming. However, due to the declarative nature of the presented algorithm, it is extensible by additional constraints. This extensibility subsumes and goes beyond earlier constrained alignment approaches. Especially, we have shown how certain prior knowledge and structure information can be incorporated into the alignment model. By applying cluster tree elimination to the resulting extended alignment problem, we solve it efficiently. Finally, we have demonstrated for the alignment problem how CTE could profit from intelligent reasoning on the constraint model. Thereby, we hint at possible improvements of a current constraint solving strategy.

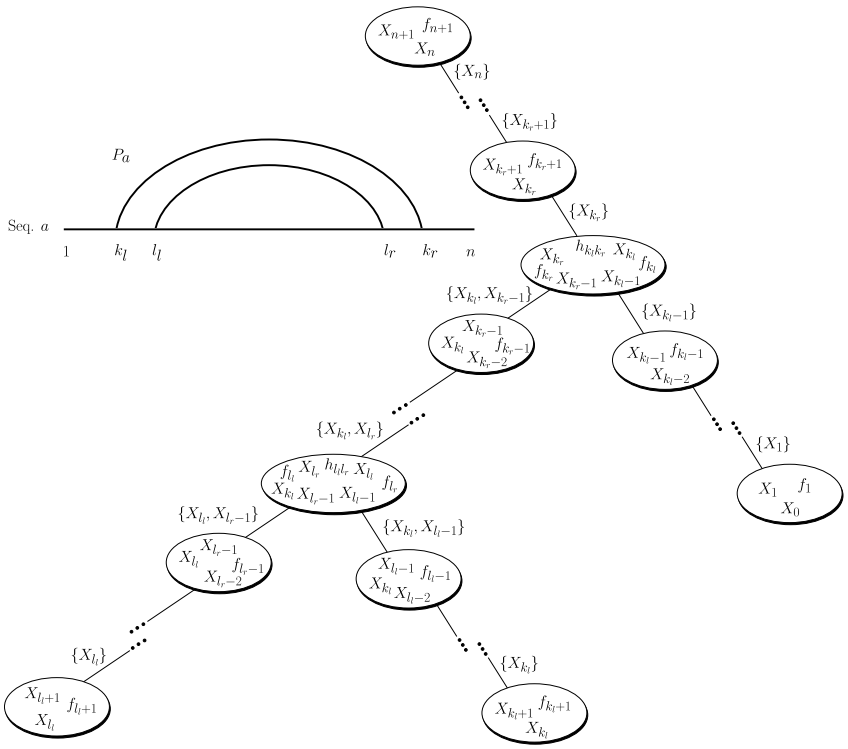

Fig. 3. Example sequence structure alignment CTD (see text for details).

References

1. Rolf Backofen and Sebastian Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2(4):681–698, 2004.
2. Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.
3. Tao Jiang, Guohui Lin, Bin Ma, and Kaizhong Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.
4. Kalev Kask, Rina Dechter, Javier Larrosa, and Avi Dechter. Unifying cluster-tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 2005. forthcoming.
5. Giuseppe Lancia, Robert Carr, Brian Walenz, and Sorin Istrail. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proc. of the Fifth Annual International Conferences on Compututational Molecular Biology (RECOMB01)*. ACM Press, 2001.
6. Hans-Peter Lenhof, Knuth Reinert, and Martin Vingron. A polyhedral approach to rna sequence structure alignment. In *Proc. of the Second Annual Interna-*

tional Conferences on Compututational Molecular Biology (RECOMB98), volume 5, pages 517–30. ACM Press, 1998.
7. Burkhard Morgenstern, Nadine Werner, Sonja J. Prohaska, Rasmus Steinkamp, Isabella Schneider, Amarendran R. Subramanian, Peter F. Stadler, and Jan Weyer-Menkhoff. Multiple sequence alignment with user-defined constraints at GOBICS. *Bioinformatics*, 21(7):1271–1273, 2005.
8. Gene Myers, Sanford Selznick, Zheng Zhang, and Webb Miller. Progressive multiple alignment with constraints. In *Proceedings of the first annual international conference on Computational molecular biology (RECOMB 1997)*, pages 220–225, 1997.
9. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–53, 1970.
10. David Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5):810–825, 1985.
11. T.F. Smith and M.S. Waterman. Comparison of biosequences. *Adv. appl. Math.*, 2:482–489, 1981.
12. Roland H. C. Yap. Parametric sequence alignment with constraints. *Constraints*, 6(2/3):157–172, 2001.