Efficient Sequence Alignment with Side-Constraints by Cluster Tree Elimination

Sebastian Will, Anke Busch and Rolf Backofen

Bioinformatics Group, Institute of Computer Science,
Albert-Ludwigs-University Freiburg
Georges-Koehler-Allee 106, 79110 Freiburg,
{*will,abusch,backofen*}*@informatik.uni-freiburg.de*

Abstract

Aligning DNA and protein sequences is a core technique in molecular biology. Often, it is desirable to include partial prior knowledge and conditions in an alignment. Going beyond prior work, we aim at the integration of such side constraints in free combination into alignment algorithms. 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. Finally, we apply our method to the scanning for RNA motifs in large sequences. This case study demonstrates how the new approach can be used in real biological problems. A prototypic implementation of the method is available at `http://www.bioinf.uni-freiburg.de/Software/CTE-Alignment`.

1 Introduction

Modern molecular biology relies heavily on 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. One example is the following. Assume that we want to align some sequences and we know already that they share certain local motifs (subsequences). Reasonably, the sequences should be compared taking this knowledge into account. Therefore, we need to optimize similarity under the additional constraint that the motifs should be matched (approximately) to each other. Another example is the enhancement of RNA (or even protein) comparison by employing knowledge on the structure of the RNAs and proteins [19, 10, 1, 21, 12]. Such tasks can get arbitrary complicated when one wants to combine different kinds of such prior knowledge in one comparison of sequences.

However in general, similarity searching tools 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 [15] and precedence constraints [16]. 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)* [17, 20, 10] or, from an algorithmic point of view, DP-based approaches like HMMs and SCFGs [18, 7]. There is no straightforward and general way to extend a DP algorithm in order to take additional knowledge into account.

In this paper, we propose a declarative formulation of the alignment problem that allows an easy integration of side constraints. In principle, declarative 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. Although there already exist other declarative alignment approaches, they were not designed to support additional side constraints. In particular, it is not obvious how these methods can be extended by biologically meaningful constraints and how this influences their efficiency. One such previous approach [13] is based on *integer linear programming (ILP)*. Since in ILP one can only use boolean variables, the ILP model of [13] 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 [22] 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.

Contribution The general problem of alignment with side constraints is NP-hard. However in many cases, adding only a few or special constraints to the alignment problem will still allow for efficient computation. Usually a constraint affects only a part of the alignment, while the rest can still be evaluated efficiently. Also, many constraints may only modify the problem but do not impair efficiency.

Here, we introduce a new constraint-based approach, which benefits of such considerations. 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 semiring-based constraint optimization problem in the sense of [3, 11] and then applying a special solution strategy, which is known as *cluster tree elimination (CTE)* [11].

Overview In the first three sections, we describe the model, depict the strategy of CTE, and then present special constraints in more detail. In Section 5, we demonstrate the use of our method for biological real-world applications by a case-study on scanning for RNA motifs. In this section, we will extend our alignment model for the special purpose of scanning and use several constraints to specify a complex sequence structure motif. We close with a discussion of our work and some open problems. In particular, we discuss a possible extension to multiple alignment with arbitrary side constraints.

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.

2.1 Sequence Alignment

We define an *alignment $\mathcal{A}$ of a and b* as a subset of pairs of positions in a and b, i.e.

$$\mathcal{A} \subset \{1, \dots, n\} \times \{1, \dots, m\},$$

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 $(i, j) \in \mathcal{A}$. Note that our definition of an alignment as a set of alignment edges (pairs of matched positions) is equivalent to the more common definition as a pair of alignment strings.

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

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

The classical DP algorithm for the alignment of a and b computes the optimal alignment score recursively from the scores $D_{i,j}$ of optimal alignments of prefixes $a_1, \dots, a_i$ and $b_1, \dots, b_j$. For computing $D_{i,j}$, it evaluates the recursion equation

$$D_{i\,j} = \max \begin{cases} D_{i-1\,j-1} + \sigma(i, j) \\ D_{i-1\,j} + \gamma \\ D_{i\,j-1} + \gamma \end{cases}$$

with initialization

$$D_{0\,0} = 0,\ D_{i\,0} = i\gamma, \text{ and } D_{0\,j} = j\gamma$$

for $1 \leq i \leq n$ and $1 \leq j \leq m$. Since the algorithm materializes intermediate values, it solves the problem of computing the alignment score $D_{n,m}$ in $O(nm)$ time.

2.2 Constraint framework

Here, we model an alignment as a constraint optimization problem in the semiring-based framework that is described in a more general form in [3, 11]. 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

$$\mathrm{dom}(X_i) = \{0, \dots, m\}.$$

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 $(X_0 = x_0, \dots, X_{n+1} = x_{n+1})$ of variables $X_0, \dots, X_{n+1}$ 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

$$\vec{x} = (0, 1, 2, 5, 6, 6, 6, 7, 8)$$

of the variables $X_0, \dots, X_8$ corresponds to the alignment

$$\{(1,1), (2,2), (3,5), (4,6), (7,7)\},$$

which can be represented alternatively by

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

Hard constraints 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

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

General and affine gap cost For simplicity, in the rest of this work we focus on alignment with linear gap cost. Nevertheless, the approach can be extended to work with arbitrary gap cost. It is even possible to integrate affine gap cost with the expected performance benefits over general gap cost [8].

Assume a non-negative function $\Gamma : \mathbb{N}^+ \to \mathbb{R}$ for gap-cost that is *sub-additive*, i.e. $\Gamma(k) + \Gamma(k') \geq \Gamma(k + k')$ $(k, k' > 0)$ and $\Gamma(0) := 0$. We are interested in modeling the score

$$\text{score}_\Gamma(\mathcal{A}) = \sum_{(i,j)\in\mathcal{A}} \sigma(i,j) + \sum_{k\geq 0} \Gamma(k) \cdot N^k_{\mathcal{A}},$$

where $N^k_{\mathcal{A}}$ is the number of gaps of length k in $\mathcal{A}$.

Due to the asymmetry of the model, arbitrary scoring of gaps in the second sequence would require only a simple modification of f_i. For scoring gaps in the first sequence too, we introduce variables G_i for $i = 0, \ldots, n+1$ with domain $\{0, \ldots, i\}$, $G_0 = 0$, and constraint functions $\text{gap}_i : \{1, \ldots, n\}^4 \to \{-\infty, 0\}$ for $i = 1, \ldots, n+1$ that express the hard constraints

$$\textbf{if } X_i = X_{i-1} \textbf{ then } G_i = G_{i-1} + 1 \textbf{ else } G_i = 0.$$

By these constraints the G_i are uniquely determined by the X_i and count the length of each gap defined by the X_i. For our example valuation, we get

i	0	1	2	3	4	5	6	7	8
X_i	0	1	2	5	6	6	6	7	8
G_i	0	0	0	0	0	1	2	0	0

Now, instead of the functions $f_i(X_{i-1}, X_i)$, one introduces new functions $\bar{f}_i(X_{i-1}, X_i, G_{i-1})$ defined by

$$\bar{f}_i(j', j, g) = \begin{cases} \Gamma(j - j' - 1) + \sigma(i,j) + \Gamma(g) & \text{if } j' < j \\ 0 & \text{otherwise.} \end{cases}$$

If Γ is an affine function, i.e. $\Gamma(k) = \alpha + \beta k$ $(k > 0)$, one can simplify the model by distinguishing gap opening and gap extension. In the model, this only simplifies the treatment of gaps in the first sequence. Taking full advantage of affine gap cost also for gaps in the second sequence requires further modifications similar to those described in Subsection 3.2.

For affine cost, extend the model with linear cost again with variables G_i and with constraint functions gap^a_i that express the constraints

$$\textbf{if } X_i = X_{i-1} \textbf{ then } G_i = 1 \textbf{ else } G_i = 0.$$

Instead of the functions $f_i(X_{i-1}, X_i)$, one introduces this time new functions $\tilde{f}_i(X_{i-1}, X_i, G_{i-1})$ defined by

$$\tilde{f}_i(j', j, g) = \begin{cases} \Gamma(j - j' - 1) + \sigma(i, j) + \alpha \cdot g & \text{if } j' < j \\ \beta & \text{otherwise.} \end{cases}$$

Now, the G_i work only as flags for gaps. The simplification for the affine case is visible in the restriction of the domain of the variables G_i to $\{0, 1\}$.

2.3 Correctness of the model

Note that we correctly model alignments and their scores. We will show this here for the model with linear gap cost. 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 the sum over all function values

$$\sum_{1 \leq i \leq n+1} f_i(x_{i-1}, x_i) + leq_i(x_{i-1}, x_i)$$

is not $-\infty$. Secondly in this case, this sum equals the alignment score due to the following proposition.

Proposition 2.1

$$\text{score}(\mathcal{A}) = \sum_{1 \leq i \leq n+1} f_i(x_{i-1}, x_i) \tag{2}$$

Proof. We show the proposition by algebraic transformation of the right hand side of Equation 2.

$$\sum_{1 \leq i \leq n+1} f_i(x_{i-1}, x_i)$$

equals by splitting the sum and case distinction (cf. definition of f_i)

$$\sum_{\substack{1 \leq i \leq n+1 \\ x_{i-1} < x_i}} \sigma(i, x_i) + \sum_{\substack{1 \leq i \leq n+1 \\ x_{i-1} < x_i}} (x_i - x_{i-1} - 1)\gamma + \sum_{\substack{1 \leq i \leq n+1 \\ x_{i-1} = x_i}} \gamma.$$

Due to the definition of our model, this is equal to

$$\sum_{(i,j) \in \mathcal{A}} \sigma(i, j) + \sum_{\substack{1 \leq i \leq n+1 \\ x_{i-1} < x_i}} (x_i - x_{i-1})\gamma - \sum_{\substack{1 \leq i \leq n+1 \\ x_{i-1} < x_i}} \gamma + \sum_{\substack{1 \leq i \leq n+1 \\ x_{i-1} = x_i}} \gamma.$$

Since $\sum_{\substack{1\le i\le n+1\\ x_{i-1}=x_i}} (x_i - x_{i-1})\gamma = 0$, this equals

$$\sum_{(i,j)\in\mathcal{A}} \sigma(i,j) + \sum_{\substack{1\le i\le n+1\\ x_{i-1}\le x_i}} (x_i - x_{i-1})\gamma - \sum_{\substack{1\le i\le n+1\\ x_{i-1}<x_i}} \gamma + \sum_{\substack{1\le i\le n+1\\ x_{i-1}=x_i}} \gamma.$$

$\{1 \le i \le n+1 \mid x_{i-1} \le x_i\}$ contains all $1 \le i \le n+1$, since $x_0, \dots, x_{n+1}$ is ordered due to the less or equal constraints. Thus, the term above equals

$$\sum_{(i,j)\in\mathcal{A}} \sigma(i,j) + (x_{n+1} - x_0)\gamma - \sum_{\substack{1\le i\le n+1\\ x_{i-1}<x_i}} \gamma + \sum_{\substack{1\le i\le n+1\\ x_{i-1}=x_i}} \gamma.$$

Since $x_{n+1} = m+1$ and $x_0 = 0$ and $\{1 \le i \le n+1 \mid x_{i-1} < x_i\}$ is the set of matched positions of sequence a in the alignment $\mathcal{A}$, unified with $\{n+1\}$, this equals

$$\sum_{(i,j)\in\mathcal{A}} \sigma(i,j) + (m+1)\gamma - (|\mathcal{A}|+1)\gamma + \sum_{\substack{1\le i\le n+1\\ x_{i-1}=x_i}} \gamma.$$

$\{1 \le i \le n+1 \mid x_{i-1} = x_i\}$ is the set of positions in a that are not matched by $\mathcal{A}$. Thus, one can transform the term into

$$\sum_{(i,j)\in\mathcal{A}} \sigma(i,j) + (m+1)\gamma - (|\mathcal{A}|+1)\gamma + (n - |\mathcal{A}|)\gamma$$

which is

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

□

3 Efficient Solving by Cluster Tree Elimination

Here, we sketch Cluster Tree Elimination (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.

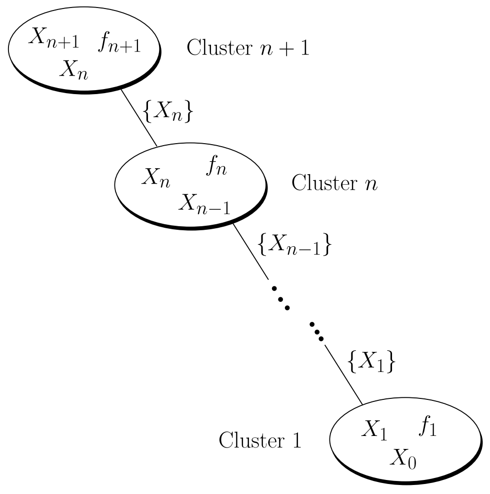

Figure 1: CTD of pure sequence alignment.

3.1 Basic Mechanism

For applying CTE, we first need a *cluster tree decomposition (CTD)* [11]. The CTD works as a guide for the solving of the constraints. The idea is that the problem can be processed by sending messages (corresponding to partial solutions) along this tree structure. 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.

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

$$\max_{1 \leq j \leq m} \left(g_n(j) + f_{n+1}(j, m+1) \right),$$

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

$$g_i(j) = \max_{0 \leq j' \leq m} \left(g_{i-1}(j') + f_i(j', j) + leq_i(j', j) \right). \tag{3}$$

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.

3.2 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 3 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 and move the constant $\sigma(i, j)$ out of the maximization. Then,

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

Proposition 3.1 *A helper function*

$$g_i^{\mathrm{m}}(j) = \max_{0 \leq j' < j} \left(g_{i-1}(j') + (j - j' - 1)\gamma \right)$$

can be defined recursively and then computed in $O(m)$ time by DP as

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

Proof. The case of $j = 1$ follows by definition. It remains to show that

$$\max_{0 \le j' < j} \big(g_{i-1}(j') + (j - j' - 1)\gamma\big) = \max \begin{cases} g_i^{\mathrm{m}}(j-1) + \gamma \\ g_{i-1}(j-1). \end{cases}$$

$$\begin{aligned} g_i^{\mathrm{m}}(j) &= \max_{0 \le j' < j} \big(g_{i-1}(j') + (j - j' - 1)\gamma\big) \\ &= \max \begin{cases} \max\limits_{0 \le j' < j-1} \big(g_{i-1}(j') + (j - j' - 1)\gamma\big) \\ g_{i-1}(j-1) + (j - (j-1) - 1)\gamma \end{cases} \\ &= \max \begin{cases} \max\limits_{0 \le j' < j-1} \big(g_{i-1}(j') + (j - 1 - j' - 1)\gamma + \gamma\big) \\ g_{i-1}(j-1) \end{cases} \\ &= \max \begin{cases} g_i^{\mathrm{m}}(j-1) + \gamma \\ g_{i-1}(j-1) \end{cases} \end{aligned}$$

□

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]

Note that the described reduction in complexity is only shown for clusters that contain exactly the variables and constraints of the (unconstrained) alignment CTD. If one adds further constraints, as described in the following, the reduction still applies to computing the messages in all unaffected clusters. In principle, the complexity reduction is possible for many types of constraints. However this requires to perform a proof for each new constraint. An example of such an optimization is given for the constraint introduced in Subsection 4.2. Without further work, the messages for modified clusters can still be computed using the default-mechanism of CTE.

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

3.3 Complexity considerations

Before we discuss concrete constraints that are useful for sequence alignment, we want to discuss the complexity of sequence alignment with CTE in a more general way.

The time complexity of CTE is $O((r+N) \cdot deg \cdot k^w)$, its space complexity $O(N \cdot k^{sep})$, where

- N is the number of vertices in the CTD
- w is the tree width of the CTD
- *sep* is the maximum size of a set of separator variables
- r is the number of functions
- *deg* is the maximum degree of the CTD
- k is the maximum domain size of a variable.

The tree width of a tree decomposition is the maximal number of variables in a cluster. For detailed definitions and theorems please see [11].

For the basic model of unconstrained sequence alignment without complexity improvement, this yields the already given complexities of $O(n \cdot m^2)$ time and $O(n \cdot m)$ space, since there $N = n + 1$, $w = 2$, $sep = 1$, $r = n + 1$, $deg = 1$, and $k = m + 2$.

Due to this analysis, if we assume that additional constraints will introduce at most $O(n)$ variables and functions, additional constraints can only increase the time complexity by increasing w and increase the space complexity by increasing *sep*.

For practical applications, we are most interested in keeping w and *sep* low. This is possible for most constraints presented in the next section, since they introduce only dependencies between variables in close distance, i.e. X_i and X_j where $|i - j|$ is small, often 1.

The single exception are the structure constraints of Subsection 4.3. Those constraints introduce long distance dependencies between variables. Introducing a long distance dependency between X_i and X_j requires a restructuring of the cluster tree, forming a cluster that contains X_i and X_j. Naturally, this leads to an increased tree width (and *sep*) compared to the basic model. Now, it is interesting to note that additional long distance interactions that do not cross the existing interactions can be added without increasing w or *sep* further. Here two interactions between i and j ($i < j$) and between i' and j' ($i' < j'$) *cross*, if either $i < i' < j < j'$ or

$i' < i < j < j'$. The phenomenon that non-crossing interactions can be handled with comparably low complexity is well known in the area of RNA alignment and tree editing. [10]

Note that our approach however is not limited to such non-crossing interactions. Quite on the opposite, the proposed method can adapt to increasingly complex interactions with increasing computational complexity. Nevertheless, we don't want to conceal here that a large number of crossing long range interactions will make the current approach impracticable. For practical applications, the possibility to include such interactions can still be valuable as long as their number is low. In this respect the approach is clearly distinguished from other dynamic programming based methods (including SCFGs [18, 7]), which cannot incorporate such interactions straightforwardly.

The presented method using CTE is therefore adapted to the case of combining (possibly many) low distance constraints and crossing long distance constraints. In this case, the tree structure adequately reflects the structure of the constrained alignment problem. When the focus is on integrating many more complex and crossing constraints, other, more advanced constraint decomposition based solving strategies might be advantageous [6, 5], which however is beyond the scope of this paper.

4 Constraints for Sequence Alignment

Recently discussed constrained alignment approaches handled constraints like precedence constraints [16] and anchor constraints [15]. 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. After reviewing these and other simple constraints, we present more complex constraints that can be handled in our framework.

4.1 Some Simple Constraints

Match A simple example of a constraint for extending the model is the *match constraint* `Match`(i, j), which enforces that a_i is matched to b_j. It restricts the class of valid alignments to those alignments $\mathcal{A}$, where

$$(i, j) \in \mathcal{A}.$$

In our model, this is expressed by adding the constraints

$$X_i = j \text{ and } X_{i-1} < X_i.$$

By definition of our model, the conjunction of these two constraints is equivalent to the match constraint. The match constraint can be extended easily to the case, where a_i must be aligned to either b_j or b'_j. The corresponding constraints are $X_i \in \{j, j'\}$ and $X_{i-1} < X_i$.

Exact Match A second simple example of a constraint that extends the model is the *exact matching constraint* $\texttt{ExMatch}(i)$. It enforces that a_i is matched to a position in b that contains the same symbol as a_i itself. Formally, the exact matching constraint $\texttt{ExMatch}(i)$ is defined on alignments $\mathcal{A}$ by

$$\exists j : (i, j) \in \mathcal{A} \wedge a_i = b_j.$$

In our model, this is expressed by adding the following constraints

$$X_i \in \{j | b_j = a_i\} \text{ and } X_{i-1} < X_i.$$

Forbidding Gaps The *no gap constraint* $\texttt{NoGap}(i, i')$ ensures that the subsequence $a_i ... a_{i'}$ is aligned to a subsequence of b without any gaps. Formally, this means that

- $(i, j) \in \mathcal{A}$ and
- $\forall 1 \leq k \leq i' - i : (i + k, j + k) \in \mathcal{A}$.

In our model, this is expressed by adding the constraints

- $X_{i-1} < X_i$ and
- $X_{i+k} = X_{i+k-1} + 1$ for all $1 \leq k \leq i' - i$.

Precedence A *precedence constraint* $\texttt{Prec}^{\texttt{L}}(i, j)$, handled in [16], tells that in the alignment position i of the first sequence is left of position j of the second sequence. Formally, this means that

$$\forall (k, l) \in \mathcal{A} : k \leq i \implies l < j.$$

In our model, this condition is encoded as

$$X_i < j.$$

Noteworthy, by the less or equal constraints leq_i of our basic constraint model, this implies for all $1 \leq k < i$ that $X_k < j$. Similarly, one can define a precedence constraint $\texttt{Prec}^{\texttt{R}}(i, j)$ that expresses the symmetric case that position i in a is right of position j in b.

Anchors An *anchor constraint*, as discussed in [15], tells that position i in the first sequence and position j in the second sequence can only be aligned to each other and furthermore, positions strictly left (resp. right) of i are aligned to positions strictly left (resp. right) of j. Formally, the anchor constraint $\texttt{Anchor}(i, j)$ is defined on alignments $\mathcal{A}$ by

$$\forall (k, l) \in \mathcal{A} : (k < i \land l < j) \lor (k > i \land l > j) \lor (k = i \land l = j)$$

In our model, this is expressed by the constraints

- $X_{i-1} < j$,
- $X_{i+1} > j$, and
- $X_i = j \lor X_i = X_{i-1}$.

Together, the first and last constraint imply $X_i \leq j$. 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 or equal constraints leq_i the new domain information is further propagated to the domains of all variables.

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.

4.2 Aligned Segment Constraints

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}, the values of A_i are in the range $\{0, \ldots, i-k+1\}$. To satisfy the constraint, A_i has to be at least $\max(0, x - (k' - i))$ since positions larger than i can contribute at most $k' - i$ matches. We can finally express the constraint by restricting the domain of A_i to

$$\{\max(0, x - (k' - i)), \ldots, i - k + 1\}.$$

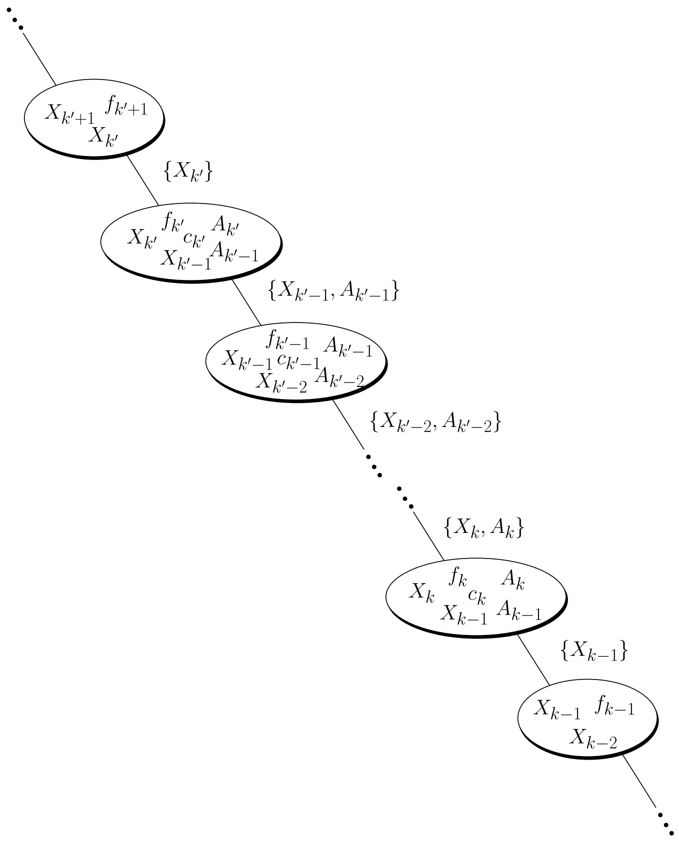
$X_{k'+1}$ $f_{k'+1}$ $X_{k'}$
$\{X_{k'}\}$
$f_{k'}$ $A_{k'}$ $X_{k'}$ $c_{k'}$ $X_{k'-1}$ $A_{k'-1}$
$\{X_{k'-1}, A_{k'-1}\}$
$f_{k'-1}$ $A_{k'-1}$ $X_{k'-1}$ $c_{k'-1}$ $X_{k'-2}$ $A_{k'-2}$
$\{X_{k'-2}, A_{k'-2}\}$
$\{X_k, A_k\}$
X_k f_k A_k c_k X_{k-1} A_{k-1}
$\{X_{k-1}\}$
X_{k-1} f_{k-1} X_{k-2}

Figure 2: CTD of an alignment with segment constraints.

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 i that contains a variable A_i (and thus contains the variables X_i, X_{i-1}, and A_{i-1} by construction). We cover only the case that the cluster $i - 1$ also contains a variable A_{i-1}, which is violated only once at the beginning of the segment. Then, 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

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

One transforms $g_i(j,a)$ further, after inserting the definition of f_i. Then, $g_i(j,a) =$

$$\max_{0\le j'\le j} \begin{cases} g_{i-1}(j',a-1) & +(j-j'-1)\gamma + \sigma(i,j) & \text{if } j' < j \text{ and } l \le j \le l' \\ g_{i-1}(j',a) & +(j-j'-1)\gamma + \sigma(i,j) & \text{if } j' < j \text{ and } j \notin [l..l'] \\ g_{i-1}(j',a) + \gamma & & \text{if } j' = j \end{cases}$$

$$= \max_{0\le j'\le j} \begin{cases} g_{i-1}(j',\mathfrak{C}(j,a)) + (j-j'-1)\gamma + \sigma(i,j) & \text{if } j' < j \\ g_{i-1}(j',a) + \gamma & \text{if } j' = j \end{cases}$$

$$= \max \begin{cases} \sigma(i,j) + \max\limits_{0\le j'<j} \big(g_{i-1}(j',\mathfrak{C}(j,a)) + (j-j'-1)\gamma\big) \\ g_{i-1}(j,a) + \gamma \end{cases}$$

where for our fixed l and l', $\mathfrak{C}(j,a) = \begin{cases} a-1 & \text{if } l \le j \le l' \\ a & \text{otherwise.} \end{cases}$

Now, we define a helper function

$$g_i^{\mathrm{m}}(j,a) = \max_{0\le j'<j} \big(g_{i-1}(j',\mathfrak{C}(j,a)) + (j-j'-1)\gamma\big).$$

Using g_i^{m}, we can express $g_i(j,a)$ by

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

Finally, g_i^{m} can be defined recursively as in the previous section as

$$g_i^{\mathrm{m}}(0,a) = -\infty,$$
$$g_i^{\mathrm{m}}(1,a) = g(1,\mathfrak{C}(j,a)),$$

and

$$\text{for } j > 1: \quad g_i^{\mathrm{m}}(j,a) = \max \begin{cases} g_i^{\mathrm{m}}(j-1,\mathfrak{C}(j,a)) + \gamma \\ g_{i-1}(j-1,\mathfrak{C}(j,a)), \end{cases}$$

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. Here, the additional constraints do not interfere with the nature of our score that enables the recursive decomposition.

4.3 Structure Constraints

Here as additional input, we have two structures (of RNAs or even proteins)

$$P_a \subset \{1,\dots,n\} \times \{1,\dots,n\}$$

and

$$P_b \subset \{1,\dots,m\} \times \{1,\dots,m\}$$

and a function

$$\omega : \{1,\dots,n\} \times \{1,\dots,n\} \times \{1,\dots,m\} \times \{1,\dots,m\} \to \mathbb{R}.$$

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

$$h_{i_l i_r}(j'_l, j_l, j'_r, j_r) = \begin{cases} \omega(i_l, i_r; j_l, j_r) & \text{if } j'_l < j_l,\ j'_r < j_r, \text{ and } (j_l, j_r) \in P_b \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Figure 3 provides an example for $P_a = \{(k_l, k_r), (l_l, l_r)\}$ and arbitrary P_b. It demonstrates the general construction principle of such a CTD for the base pairs in P_a that are not crossing. Only for non-crossing base-pairs the tree width and separator size can be controlled in this way, crossing base pairs introduce crossing dependencies (cf the discussion of Subsection 3.3 for crossing interactions). The good support for non-crossing base pairs favors the use for completely or almost non-crossing RNA structures.

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

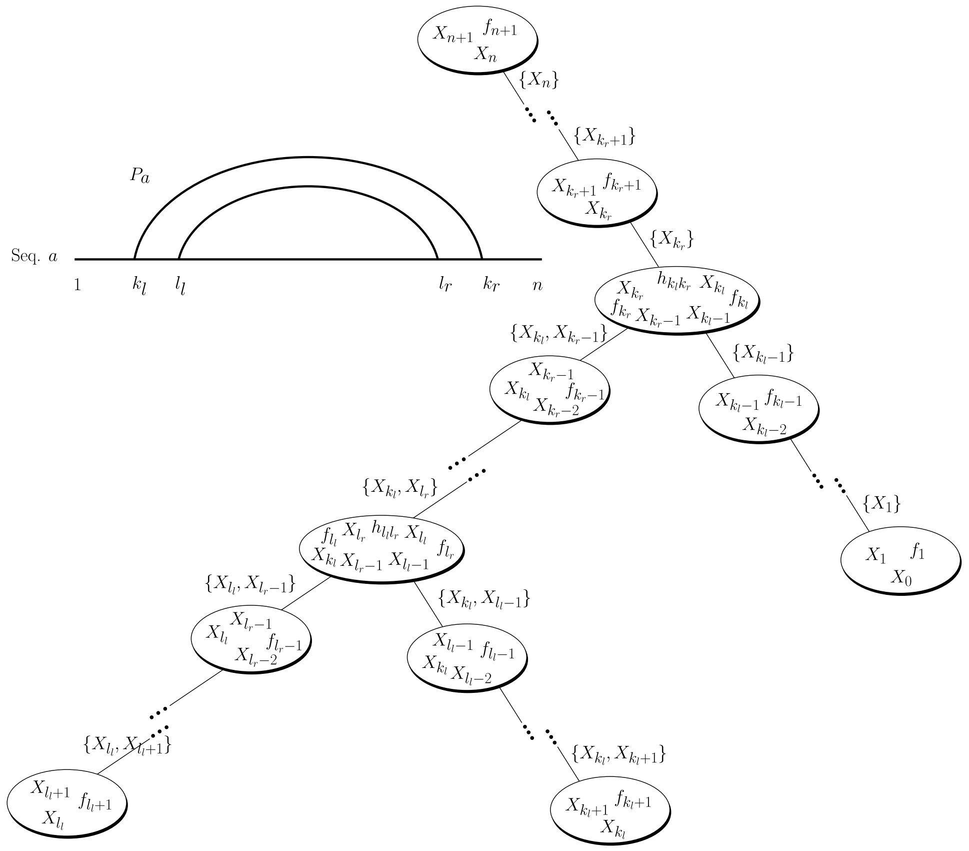
X_{n+1} f_{n+1} X_n
$\{X_n\}$
$\{X_{k_r+1}\}$
X_{k_r+1} f_{k_r+1} X_{k_r}
P_a
Seq. a
1 k_l l_l l_r k_r n
$\{X_{k_r}\}$
X_{k_r} $h_{k_lk_r}$ X_{k_l} f_{k_l} f_{k_r} X_{k_r-1} X_{k_l-1}
$\{X_{k_l}, X_{k_r-1}\}$
$\{X_{k_l-1}\}$
X_{k_r-1} X_{k_l} f_{k_r-1} X_{k_r-2}
X_{k_l-1} f_{k_l-1} X_{k_l-2}
$\{X_{k_l}, X_{l_r}\}$
$\{X_1\}$
f_{l_l} X_{l_r} $h_{l_ll_r}$ X_{l_l} f_{l_r} X_{k_l} X_{l_r-1} X_{l_l-1}
X_1 f_1 X_0
$\{X_{l_l}, X_{l_r-1}\}$
$\{X_{k_l}, X_{l_l-1}\}$
X_{l_r-1} X_{l_l} f_{l_r-1} X_{l_r-2}
X_{l_l-1} f_{l_l-1} X_{k_l} X_{l_l-2}
$\{X_{l_l}, X_{l_l+1}\}$
$\{X_{k_l}, X_{k_l+1}\}$
X_{l_l+1} f_{l_l+1} X_{l_l}
X_{k_l+1} f_{k_l+1} X_{k_l}

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

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 Case Study: Scanning for RNA Motifs

For demonstrating the use of our method in real biological applications, we study the scanning for non-coding RNA (ncRNA). In this problem, we want to find a similar occurrence of a given ncRNA in a larger sequence, e.g. a complete genome. This scanning must take into account sequence and structure, and additionally satisfy certain constraints.

An example of such an ncRNA is small nucleolar RNA (snoRNA). It has a characteristic structure, see Fig 4, consisting of two stems and two highly conserved sequence boxes ANANNA and ACA (H/ACA-type snoRNA). Here, the N in the sequence ANANNA is the usual symbol for any nucleotide; a stem is the common structural motif of RNAs consisting of a series of stacked base pairs. We can describe the search pattern by giving the structure as a set of base pairs and the two sequence boxes with their positions in the pattern. Hence, we need to combine three kinds of constraints in this example, namely

- structure constraints,
- exact match constraints, and
- no gap constraints.

This combination will not be easily expressed in other settings. In the following, we extend our constraint model for the special purpose of scanning (Subsection 5.1). Then we discuss the integration of side constraints in Subsection 5.2.

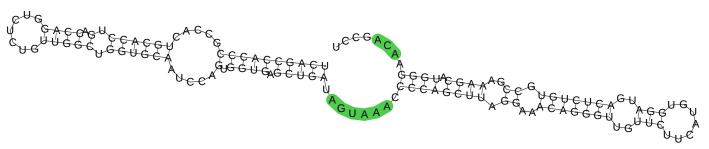

Figure 4: A snoRNA with its characteristic structure and highly conserved sequence boxes.

5.1 A scanning constraint model

For searching such a pattern in a large target sequence, we first need to modify our basic constraint model for sequence alignment.

For explaining our extension, we use a running example, where we scan for a pattern of length 7, i.e. sequence a in a large sequence b, here of length 100. A possible alignment for this example could look like this:

$$\begin{array}{ccccccccccccc} & & & a_1 & a_2 & a_3 & - & a_4 & a_5 & a_6 & a_7 & & \\ b_1 & \dots & b_{50} & - & - & b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & - & b_{56} \dots & b_{100}, \end{array} \tag{5}$$

i.e. $\mathcal{A} = \{(3,51),(4,53),(5,54),(6,55)\}$.

When scoring this alignment for scanning purposes, the gaps to the left and right of a must not contribute to the score. Therefore, we modify the cost functions f_i in order to allow such gaps without cost. The functions are re-defined as

$$\text{for } 1 \le i \le n: \qquad f_i(j',j) = \begin{cases} \sigma(i,j) & \text{if } j' = 0 \text{ and } j' < j \\ (j - j' - 1)\gamma + \sigma(i,j) & \text{if } j' \neq 0 \text{ and } j' < j \\ \gamma & \text{otherwise} \end{cases}$$

$$\text{and} \qquad f_{n+1}(j',j) = 0.$$

Then, our scanning example can be modeled using variables $X_0, \ldots, X_8$, the constraints leq_i and the modified functions f_i. Our example alignment of Eq. (5) corresponds to the valuation

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
0	0	0	51	53	54	55	55	101

Now, the valuation is scored by the sum of function values

$$f_1(0,0) + f_2(0,0) + f_3(0,51) + f_4(51,53) + \\ f_5(53,54) + f_6(54,55) + f_7(55,55) + f_8(55,101)$$

and due to our modification in the scanning model this is evaluated as

$$= \gamma + \gamma + \sigma(3,51) + [\gamma + \sigma(4,53)] + \sigma(5,54) + \sigma(6,55) + \gamma + 0 \\ = \sum_{(i,j)\in\mathcal{A}} \sigma(i,j) \quad + \quad 4\gamma.$$

As intended, the new scoring does not penalize the end gaps in the first alignment row.

5.2 Constraints for ncRNA scanning

We can now extend the scanning model for our special application of snoRNA finding by adding constraints. We denote the snoRNA sequence of length n by a and the set of base pairs in our snoRNA pattern by P_a, b is our target sequence with length m.

Constraining Sequence Boxes First, we need to enforce that the sequence boxes are matched exactly and without gaps to identical sequences in b. We add the following constraints for the ANANNA box at positions $66 - 71$ in sequence a (cf. Subsection 4.1 for the precise interpretation of the constraints).

- `ExMatch`(i) for $i = 66, 68, 71$
- `NoGap`$(66, 71)$.

and for the ACA box (positions 132-134) the constraints

- `ExMatch`(i) for $i = 132, 133, 134$
- `NoGap`$(132, 134)$.

These constraints can be added to our model one by one. The resulting cluster tree decomposition can be computed incrementally while inserting the constraint. Each time, we will add constraints to certain nodes of the CTD, however none of these constraints requires to change the structure of the CTD.

Scoring structure More complex is the scoring of structural similarity. For our case study, we choose to score base pair matches using base pair probabilities. There, we adapt an idea of PMcomp [9], which was also pursued in [21] and proved biologically meaningful. There is even earlier work using pair probabilities in the context of constraints [4].

Following the idea of [9, 21], we predict the probability p_{ij} of each base pair (i, j) in structures of b, using a local variant of McCaskill's algorithm [14, 2]. Note that these probabilities reflect a full energy model of RNAs. Then, each match of base pairs (i, j) in a and (k,l) in b shall improve the score by

$$\omega(i, j; k, l) = \tau \log \frac{p_{kl}}{p_0}$$

with a suitable choice of p_0 as "background probability". τ is just a weighting factor to balance the score contribution of sequence and structure similarity.

This allows us to score any base pair match and thus we set

$$P_b = \{(k, l) | 1 \leq k < l \leq m, p_{kl} > 0\}.$$

This scoring will reflect the structural similarity of the pattern and the target sub-sequence sufficiently well, without the need of a single, fixed structure in the target sequence.

Then, we extend our model by adding the constraint of Eq. 4 in Subsection 4.3 for each base pair $(i_l, i_r) \in P_a$. Practically, we add the constraints iteratively to our model. Each time, we will also compute the new cluster tree decomposition. Importantly, this requires a re-structuring of the CTD for each constraint. This constraint specific rearrangement can be given generically for the newly introduced dependencies.

6 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. Furthermore, 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. As a demonstration of the CTE-based alignment method, we provide a prototypic implementation (`http://www.bioinf.uni-freiburg.de/Software/CTE-Alignment`).

It is therefore still attractive to study constrained alignment using more advanced constraint solving techniques like WCSP with tree decompositions [5] or decomposition with branch-and-bound, e.g. AND/OR search [6]. Also improved techniques could promote the automation of the method and make even more complex combinations of side constraints practicable.

The biological relevance and use of the presented work is demonstrated by a case study about motif scanning. There, we discuss how to find similar occurrences of snoRNA sequence structure patterns in large sequences.

An interesting further research direction is the extension of the introduced approach to multiple alignment. There, in particular the extension via the progressive alignment heuristic seems promising. Important issues there are the combination of constraints and how to guarantee that for all possible consistent constraint sets an alignment can be constructed.

Acknowledgment

We thank the anonymous reviewers who helped to improve the quality of the paper. Rolf Backofen and Sebastian Will are partially supported by the

EU Network of Excellence REWERSE (project reference number 506779). We thank Jana Hertel and Peter Stadler for motivating the ncRNA-scanning example.

References

[1] R. Backofen and S. Will. Local sequence-structure motifs in RNA. *Journal of Bioinformatics and Computational Biology (JBCB)*, 2(4):681–698, 2004.

[2] S. H. Bernhart, I. L. Hofacker, and P. F. Stadler. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–5, 2006.

[3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.

[4] F. Chetouani, P. Monestie, P. Thebault, C. Gaspin, and B. Michot. ESSA: an integrated and interactive computer tool for analysing RNA secondary structure. *Nucleic Acids Research*, 25(17):3514–22, 1997.

[5] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting tree decomposition and soft local consistency in weighted csp. In *Proc. of AAAI-06*, page 6, 2006.

[6] R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.

[7] S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.

[8] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[9] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 2004.

[10] T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between RNA structures. *Journal of Computational Biology*, 9(2):371–88, 2002.

[11] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying cluster-tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166(1-2):165–193, 2005.

[12] G. Lancia, R. Carr, B. Walenz, and S. 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.

[13] H.P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. In *Proc. of the Second Annual International Conferences on Compututational Molecular Biology (RECOMB98)*, volume 5, pages 517–30. ACM Press, 1998.

[14] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–19, 1990.

[15] B. Morgenstern, N. Werner, S. J. Prohaska, R. Steinkamp, I. Schneider, A. R. Subramanian, P. F. Stadler, and J. Weyer-Menkhoff. Multiple sequence alignment with user-defined constraints at GOBICS. *Bioinformatics*, 21(7):1271–1273, 2005.

[16] G. Myers, S. Selznick, Z. Zhang, and W. Miller. Progressive multiple alignment with constraints. In *Proceedings of the first annual international conference on Computational molecular biology (RECOMB 1997)*, pages 220–225, 1997.

[17] 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.

[18] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjolander, R. C. Underwood, and D. Haussler. Recent methods for RNA modeling using stochastic context-free grammars. In *Proc. 5th Symp. Combinatorical Pattern Matching*, 1994.

[19] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5):810–825, 1985.

[20] T.F. Smith and M.S. Waterman. Comparison of biosequences. *Adv. appl. Math.*, 2:482–489, 1981.

[21] S. Will, K. Reiche, I. L. Hofacker, P. F. Stadler, and R. Backofen. Inferring non-coding rna families and classes by means of genome-scale structure-based clustering. *PLOS Computational Biology*, 3(4):e65, 2007.

[22] R. H. C. Yap. Parametric sequence alignment with constraints. *Constraints*, 6(2/3):157–172, 2001.