Alignment of RNA with Structures of Unlimited Complexity

Alessandro Dal Palù
Dipartimento di Matematica
Università Parma, Parma, Italy
alessandro.dalpalu@unipr.it

Mathias Möhl
Bioinformatics, Institute of Computer Science
Albert-Ludwigs-Universität, Freiburg, Germany
mmohl@informatik.uni-freiburg.de

Sebastian Will
Biology Lab, CSAIL
MIT, Cambridge MA, USA
swill@csail.mit.edu

Abstract

Sequence-structure alignment of RNA with arbitrary secondary structure is Max-SNP-hard. Therefore, the problem of RNA alignment is commonly restricted to nested structure, where dynamic programming yields efficient solutions. However, nested structure cannot model pseudoknots or even more complex structural dependencies. Nevertheless those dependencies are essential and conserved features of many RNAs. Only a few existing approaches deal with crossing structures. Here, we present a constraint approach for alignment of structures in the even more general class of unlimited structures. Our central contribution is a new RNA alignment constraint propagator. It is based on an efficient $O(n^2)$ relaxation of the RNA alignment problem. Our constraint-based approach Carna solves the alignment problem for sequences with given input structures of unlimited complexity. Carna is implemented using Gecode.

In the post-genomic era, biologists get more and more interested in studying non-coding RNA molecules with catalytic and regulatory activity as central players in biological systems. The computational analysis of non-coding RNA requires to take structural information into account. Whereas RNAs form three-dimensional structures, structural analysis of RNA is usually concerned with the secondary structure of an RNA, i.e. the set of RNA base pairs (i,j) that form contacts (H-bonds) between the bases i and j. The RNA alignment problem is to align two RNA sequences A and B with given secondary structure for each RNA such that a score based on sequence and structure similarity is optimized. The difficulty of this problem depends on the complexity of the RNA structures. Therefore, a complexity hierarchy of RNA structures was introduced. Most RNA analysis is performed for the class of nested structures P, where base-pairs do not cross, because for this class one can find efficient dynamic programming algorithms for structure prediction and alignment under reasonable scoring schemes [13, 6]. The more general class of crossing RNA structures P restricts the degree of base pairing to at most one, as is commonly assumed for single RNA structure. Prediction and alignment in this class is NP-hard in general [2]. However, one can devise a number of algorithms that efficiently predict or align RNAs with structures from classes in between non-crossing and arbitrary crossing [10, 9, 8]. However these algorithms have complexities that limit their application range. Other approaches for RNA alignment handle crossing structures with parametrized complexity, were the parameter captures the complexity of the structures [7]. Finally, the ILP approach Lara [1] computes alignments of arbitrarily complex crossing structures and appears to be more effective than dynamic programming based approaches. The success of this AI technique was a strong motivation for this work, where we study the alignment of RNAs with structures of unlimited complexity using constraint programming.

Contribution We devise a constraint algorithm for the problem of aligning two RNA molecules with given sequences and unlimited secondary structures. By modeling and propagating constraints on integers, the method goes beyond rephrasing the ILP approach [1] in CP. We describe the constraint model, develop a new RNA alignment propagator, and present a specific search strategy. It is implemented using the Gecode constraint programming system. Finally, we apply our method to align both RNA molecules with given fixed structures and RNA molecules with associated base pair probability matrices.

1 Methods

1.1 Preliminaries

An *RNA sequence* S is a string over the set of bases $\{A,U,C,G\}$ and an *RNA structure* P is a set of *base pairs* (also called *arcs*) (i,j) with $1 \leq i < j \leq |S|$. We define an *arc-annotated sequence* as pair of RNA sequence and RNA structure and denote the i-th symbol of S by $S[i]$.

One constructs a hierarchy of RNA structure classes based on the following properties. Two arcs (i,j) and (i',j') are called *nested* iff $i < i' < j' < j$ or $i' < i < j < j'$, they are *independent* iff $i < j < i' < j'$ or $i' < j' < i < j$. A RNA structure P is called *nested* if all differing base pairs $(i,j),(i',j') \in P$ are either nested or independent. In a *crossing* RNA structure P each base is involved in at most one base pair, i.e. $\forall (i,j) \neq (i',j') \in P : i \neq i' \wedge j \neq j' \wedge i \neq j \wedge i' \neq j'$. We use the term *unlimited* to refer to an arbitrary RNA structure. Note that by definition each nested structure is crossing, and each crossing structure is unlimited, such that these classes form a class hierarchy.

An *alignment* A of two *arc-annotated sequences* (S_a,P_a) and (S_b,P_b) is a set $A_m \cup A_g$, where $A_m \subseteq [1..|S_a|] \times [1..|S_b|]$ is a set of *match edges* such that for all $(i,j),(i',j') \in A_m$ it holds that 1.) $i > i'$ implies $j > j'$ and 2.) $i = i'$ if and only if $j = j'$ and A_g is the set of *gap edges* $\{\,(x,-) \mid x \in [1..|S_a|] \wedge \nexists y : (x,y) \in A_m\,\} \cup \{\,(-,y) \mid y \in [1..|S_b|] \wedge \nexists x : (x,y) \in A_m\,\}$. We define the (i,i')*-prefix of* A as $A \cap (\{\,(j,j') \mid j \leq i, j' \leq i'\,\} \cup \{\,(j,-) \mid j \leq i\} \cup \{\,(-,j') \mid j' \leq i'\,\})$ and the (i,i')*-suffix of* A as $A \cap (\{\,(j,j') \mid j > i, j' > i'\,\} \cup \{\,(j,-) \mid j > i\} \cup \{\,(-,j') \mid j' > i'\,\})$.

Fix two arc-annotated sequences (S_a,P_a) and (S_b,P_b) with unlimited structures P_a and P_b. Define the score of alignment A of (S_a,P_a) and (S_b,P_b) as

$$\operatorname{score}(A_m \cup A_g) := \sum_{(i,i') \in A_m} \sigma(i,i') + \sum_{\substack{(i,j) \in P_a, (i',j') \in P_b,\\ (i,i') \in A_m, (j,j') \in A_m}} \tau(i,j,i',j') \quad + \quad \gamma |A_g|,$$

where $\sigma(i,j)$ is the similarity of bases $S_a[i]$ and $S_b[j]$, $\tau(i,j,i',j')$ is the similarity of base pairs $(i,j) \in P_a$ and $(i',j') \in P_b$ and γ is the gap cost. Commonly, scores for sequence-structure alignment penalize the base match of different bases but don't penalize the same match if it occurs as part of a base pair match. We emphasize that our scoring function can express such scores, in the same way as scoring functions that don't add base similarity in case of a structural base match. For example, if bases A_i and B'_i differ, the negative contribution by $\sigma(i,i')$ can be compensated by $\tau(i,i',j,j')$ in a structural match.

The *alignment problem* is to determine $\displaystyle\operatorname*{argmax}_{A \text{ alignment of } (S_a,P_a) \text{ and } (S_b,P_b)} \operatorname{score}(A)$.

Please note that we score the matches of all base pairs that are matched by the alignment. Given unlimited structures P_a and P_b, our approach is thus able to simultaneously take into account several biologically relevant RNA structures per sequence. In contrast, Lara [1] would select a single, best crossing RNA structure for each sequence and score the match of only those structures. This assumes that there is only one conserved crossing structure for each RNA. The potential advantages of our scoring for aligning RNAs with conserved unlimited structure have still to be explored (see Discussion). For the special case of crossing structures with positive weights there is no difference between the scoring by our approach and Lara, because in this case Lara scores the matches of all base pairs matched by the alignment. This justifies our direct comparison of the two approaches for this case.

1.2 Constraint Model

We model an alignment of arc-annotated sequences (S_a,P_a) and (S_b,P_b) by variables $\mathtt{MD}_i$ and $\mathtt{M}_i$ for $1 \leq i \leq |S_a|$ with initial domains $D(\mathtt{MD}_i) = \{1,\ldots,|S_b|\}$ and $D(\mathtt{M}_i) = \{0,1\}$. We write $\vec{MD}$ and $\vec{M}$ to denote the vectors of respective variables $\mathtt{MD}_i$ and $\mathtt{M}_i$.

A valuation V of these variables corresponds to a class $\mathscr{A}(V)$ of alignments A of (S_a,P_a) and (S_b,P_b) as defined by

$$\begin{aligned} &V(\mathtt{MD}_i) = j \wedge V(\mathtt{M}_i) = 1 \text{ iff } (i,j) \in A \\ &V(\mathtt{MD}_i) = j \wedge V(\mathtt{M}_i) = 0 \text{ iff } (i,-) \in A \\ &\qquad\qquad\qquad \wedge \forall (i',j') \in A : i' < i \rightarrow j' \leq j \wedge i' > i \rightarrow j' > j. \end{aligned}$$

In this way, M_i tells whether i is matched or deleted and the value j of MD_i tells that i is matched to j or deleted after j. One can show that $\mathscr{A}(V)$ has at most one element and that for each alignment A of (S_a, P_a) and (S_b, P_b) there is a corresponding valuation.

For example, the following alignment and valuation correspond to each other:

$$A = \{(1,1), (-,2), (-,3), (2,4), (3,-), (4,5)\}$$

, which is often written as $\begin{matrix}\texttt{A--CUG}\\ \texttt{ACAC-G}\end{matrix}$, corresponds to the valuation $\vec{MD} = (1,4,4,5)$ and $\vec{M} = (1,1,0,1)$.

Notably, alignments corresponding to a valuation that assigns $MD_i = j$ can be composed from an alignment of prefixes $S_a[1..i]$ and $S_b[1..j]$ and an alignment of suffixes $S_a[i+1..|S_a|]$ and $S_b[j+1..|S_b|]$ regardless of $\mathtt{M}_i$.

We introduce a constraint $\mathtt{Alignment}(\vec{MD}, \vec{M})$ that is satisfied by any valuation with a corresponding alignment. Furthermore, we model the score of the alignment. Therefore, we introduce a variable `Score` and a constraint $\mathtt{AlignmentScore}(\vec{MD}, \vec{M}, \mathtt{Score})$. This constraint relates a valuation of `MD` and `M` to the score of its corresponding alignment.

Both constraints are propagated by the propagator of the next subsection. For finding optimal alignments we perform a depth-first branch-and-bound search enumerating `MD` and `M` according to a specific search strategy described at the end of the next section. Successfully applying branch-and-bound requires good upper bounds for the alignment score, such that large parts of the search tree can be pruned. Computing such bounds efficiently is the central job of the alignment propagator.

1.3 The Alignment Propagator

The alignment propagator computes hyper-arc consistency for the constraint $\mathtt{Alignment}(\vec{MD}, \vec{M})$ and propagates $\mathtt{AlignmentScore}(\vec{MD}, \vec{M}, \mathtt{Score})$.

It prunes $\vec{MD}$ and $\vec{M}$ due to the score by computing upper score bounds for single variable assignments and furthermore computes lower and upper bounds for `Score` based on $\vec{MD}$ and $\vec{M}$.

Define the class $\mathscr{A}(D)$ as union of $\mathscr{A}(V)$ over all valuations V that satisfy D. The computation of bounds is based on a relaxation of the alignment problem. In this relaxation the two ends of each base pair match are decoupled. Thus in the *relaxed optimization problem for D*, we maximize a relaxed score

$$\mathrm{score}_{\mathrm{relaxed}}(A_m \cup A_g) := \sum_{(i,i') \in A_m} \left[\sigma(i,i') + \frac{1}{2}\mathrm{ub}_D(i,i') \right] \quad + \quad \gamma |A_g|,$$

over all alignments in $\mathscr{A}(D)$, where

$$\mathrm{ub}_D(i,i') := \max_{A_m \cup A_g \in \mathscr{A}(D)} \sum_{\substack{(i,j) \in P_a, (i',j') \in P_b,\\ (i,i') \in A_m, (j,j') \in A_m}} \tau(i,j,i',j') \;+ \sum_{\substack{(j,i) \in P_a, (j',i') \in P_b,\\ (i,i') \in A_m, (j,j') \in A_m}} \tau(j,i,j',i').$$

Here, ub_D works as an upper bound for the score contributions by arc matches involving (i,i') and consequently $\mathrm{score}_{\mathrm{relaxed}}(A) \geq \mathrm{score}(A)$ for $A \in \mathscr{A}(D)$. Thus, solving the relaxed problem yields an upper bound of `Score`.

For a moment, postpone how to efficiently compute $\mathrm{ub}_D(i,i')$. Then, because the relaxed score has the form of a sequence similarity score, one can apply the Smith-Waterman algorithm [11] to maximize the relaxed score in $O(n^2)$ by dynamic programming, where $n = \max(|S_a|, |S_b|)$. The optimization problem is easily constrained due to domain D, because domains directly restrict the valid cases in the dynamic programming recursion.

Tracing back through the dynamic programming matrix yields an alignment A^l such that $\mathrm{score} A^l$ is a lower bound of `Score`. Furthermore, we compute upper bounds for each single variable valuation. This requires to complement the above "forward algorithm" that computes the matrix entries

$$\mathrm{Prefix}\,(i,i') := \max_{(i,i')\text{-prefix } A^p_{ii'} \text{ of } A \in \mathscr{A}(D)} \mathrm{score}_{\mathrm{relaxed}}(A^p_{ii'})$$

by a symmetric "backward algorithm" that computes the entries

$$\mathrm{Suffix}\,(i,i') := \max_{(i,i')\text{-suffix } A^s_{ii'} \text{ of } A \in \mathscr{A}(D)} \mathrm{score}_{\mathrm{relaxed}}(A^s_{ii'}).$$

Now the variables $\vec{MD}$ can be pruned efficiently, because Prefix $(i,i') +$ Suffix (i,i') is an upper bound for the assignment $\mathtt{MD}_i = j$. Similarly, we prune $\vec{M}$ using the two matrices.

It remains to describe the efficient computation of $\mathrm{ub}_D(i,i')$. It suffices to describe the maximization of $\sum_{\substack{(i,j)\in P_a,(i',j')\in P_b,\\(i,i')\in A_m,(j,j')\in A_m}} \tau(i,j,i',j')$ over alignments in $\mathscr{A}(D)$. A single match (j,j') can occur in an alignment in $\mathscr{A}(D)$ if $j' \in D(\mathtt{MD}_j)$ and $1 \in D(\mathtt{M}_j)$. However, we look for the best set of simultaneously valid matches (j,j'). The structure of this subproblem is analogous to sequence alignment. Thus, it is solved efficiently by dynamic programming. Therefore, $\mathrm{ub}(i,i')$ is computed in $O(kk')$ time, where k and k' are the respective number of base pairs incident to i and i'. For many applications k and k' can be constantly bounded such that the propagator runs in $O(n^2)$ time and space.

Incrementality The propagator profits from reduced domain sizes of the variables $\vec{MD}$, because Prefix (i,i') is finite only if $i' \in D(\mathtt{MD}_i)$ and the Suffix-matrix is analogously restricted. The complexity of the propagator is therefore given more precisely as $O(\sum_{i=1}^{|S_a|} |D(\mathtt{MD}_i)|)$. We postponed the idea of incrementally updating the matrices according to domain changes, because we expect large domain changes due to our propagator. Large domain changes would likely counteract the benefits of matrix updates.

Affine gap cost The method is straightforwardly extended to affine gap cost by using a Gotoh-like forward and backward algorithm in the propagator without increasing its complexity. It appears that this modification comes more natural in our approach than the corresponding extension in ILP, because it does not require any change of the model.

Propagator-guided search strategy Our search strategy guides the search to disprove overestimated bounds fast and to find valid good alignments quickly. Because information for achieving both goals is computed during propagation and is expensive to recompute, we reuse propagation results for guiding the search. We select a variable with large domain size that yields a high undecided contribution to the upper bound. We split the domain of this variable to select the 20% highest relaxed scores first.

2 Results

The method, called Carna, is implemented in C++ using the constraint programming system Gecode. For handling input and output as well as for special datastructures we reused code of LocARNA [12].

We run tests for two application scenarios. First, we explore Carna's behavior on crossing input structure using instances from all 16 Rfam families with crossing structure. Table 1 compares our results to Lara [1]. The table omits all 8 instances where both approaches run in less than 0.1 seconds. In all but one of the omitted cases, Carna solves the problem without backtracking. In terms of performance, with the single exception of tmRNA, both programs are on a par.

In our second scenario, we align dot-plot matrices as computed by RNAfold[5]. We obtain unlimited input structures by base pair filtering as e.g. done in LocARNA. As in LocARNA and PMcomp [4] base pair similarities are weighted according to the base pair probabilities. This results in alignments that are guided by the common structural potential of both RNAs and not only a single common structure. We align two tRNAs closing the search tree after nine nodes. Two TPP riboswitches of sizes 108 and 111 are aligned in 0.24 seconds closing the tree after 100 nodes.

3 Discussion

We showed that a constraint-based approach to RNA alignment can be competitive with the ILP based method Lara for crossing structures. Moreover, the approach is the first such method that scores unlimited structure. In this way, it differs from simultaneous alignment and folding approaches like Lara, which score only crossing (or even more restricted) substructures of unlimited input structures. The full potential of scoring unlimited structure and its biological applications, e.g. for aligning dot-plots of riboswitches and RNAs with conserved

Family	Instance Size				Run-time (s)		Carna Search Tree		
	$\|S_a\|$	$\|S_b\|$	$\|P_a\|$	$\|P_b\|$	Carna	Lara	Depth	Fails	Size
Entero_OriR	126	130	35	41	0.03	0.18	38	13	50
Intron_gpI	443	436	60	60	0.1	0.2	0	0	1
IRES_Cripavirus	202	199	59	57	0.2	0.04	157	127	296
RNaseP_arch	303	367	88	110	0.46	1.4	63	8	64
RNaseP_bact_b	408	401	125	125	3.0	2.3	370	677	1463
RNaseP_nuc	317	346	65	66	0.07	2.9	14	4	16
Telomerase-vert	448	451	112	116	0.47	2.3	146	32	161
tmRNA	384	367	110	110	63	3.7	433	14347	28785

Table 1: Results for the eight hardest instances of the benchmark set with crossing structures. We omit details for 8 instances where both programs run in less than 0.1 seconds.

folding dynamics have still to be explored. A constraint-based method promises flexibility for further extensions and improvements. Solving relaxed problems in propagators for handling crossing and unlimited RNA structure was shown to be a viable approach and appears to be generalizable to related problems.

Acknowledgments This work is partially supported by DFG grants WI 3628/1-1, BA 2168/3-1, PRIN08 *Innovative multi-disciplinary approaches for constraint and preference reasoning* and GNCS-INdAM *Tecniche innovative per la programmazione con vincoli in applicazioni strategiche*.

References

[1] Markus Bauer, Gunnar W. Klau, and Knut Reinert. Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. *BMC Bioinformatics*, 8:271, 2007.

[2] Guillaume Blin, Guillaume Fertin, Irena Rusu, and Christine Sinoquet. Extending the hardness of RNA secondary structure comparison. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, First International Symposium, ESCAPE 2007, Hangzhou, China, April 7-9, 2007, Revised Selected Papers*, volume 4614 of *Lecture Notes in Computer Science*, pages 140–151. Springer, 2007.

[3] Patricia A. Evans. Finding common subsequences with arcs and pseudoknots. In *CPM '99: Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching*, pages 270–280, London, UK, 1999. Springer-Verlag.

[4] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 20(14):2222–7, 2004.

[5] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie*, 125:167–188, 1994.

[6] 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.

[7] Mathias Möhl, Sebastian Will, and Rolf Backofen. Fixed parameter tractable alignment of RNA structures including arbitrary pseudoknots. In *Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching (CPM 2008)*, LNCS, pages 69–81. Springer-Verlag, 2008.

[8] Mathias Möhl, Sebastian Will, and Rolf Backofen. Lifting prediction to alignment of RNA pseudoknots. *Journal of Computational Biology*, 2010. Accepted.

[9] Jens Reeder and Robert Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, 2004.

[10] E. Rivas and S. R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, 285(5):2053–68, 1999.

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

[12] Sebastian Will, Kristin Reiche, Ivo L. Hofacker, Peter F. Stadler, and Rolf Backofen. Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLOS Computational Biology*, 3(4):e65, 2007.

[13] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–48, 1981.