Vol. 00 no. 00 2009
Pages 1–6

Constraint-based strategy for pairwise RNA secondary structure prediction

Olivier Perriquet*, Pedro Barahona
CENTRIA - Centre for Artificial Intelligence - Departamento de Informática, FCT/UNL Quinta da Torre 2829-516 CAPARICA - Portugal Tel. (+351) 21 294 8536 FAX (+351) 21 294 8541

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Context: RNA secondary structure prediction depends on the situation: when only a few (sometimes putative) RNA homologs are available, one of the most famous approach is based on a set of recursions proposed by Sankoff in 1985. Although this modus operandi insures an algorithmically optimal result, the main drawback lies in its prohibitive time and space complexities. A series of heuristics were developed to face that difficulty and turn the recursions usable.
Results: In front of the inescapable intricacy of the question when handling the full thermodynamic model, we come back in the present paper to a biologically simplified model that helps us focusing on the algorithmic issues we want to overcome. We expose our findings and forthcoming developments by using the constraint paradigm which we believe is a powerful framework for heuristic design. By doing so, we show that the main heuristics proposed by others (structural and alignment banding, multi-loop restriction) can be refined in order to produce a substantial gain both in time computation and space requirements. An implementation of our approach, named ARNICA, exemplify that gain, specially on sample sets that often remain unaffordable to other methods.
Availability: The sources and sample tests of ARNICA are available at http://centria.di.fct.unl.pt/~op/arnica.tar.gz
Contact: olivier@perriquet.net

INTRODUCTION

The interest for RNA secondary structure prediction in the last decades may be driven by the combination of two reasons, a biological one and a theoretical one. From the biological viewpoint, RNA secondary structure prediction is a challenging problem to help bridging the gap between the different levels of description of the structure (Marti-Renom *et al.*, 2008; Shapiro *et al.*, 2007). The discovery of new families of non-coding RNA (ncRNA) demands adapted tools to predict or at least provide information about their structure, and the programs that compute secondary structure naturally organize themselves along several noticeable directions that implicitly depend on their context of use. When the family of sequences under consideration is large enough and if the sequences are not too divergent, then hopefully a good multiple alignment can be reached by sequence alignment methods ((O'Brien & Higgins, 1998) assess the reliability of such methods for rRNA) or by semi-automated methods. The structure prediction programs based on covariation analysis (Eddy & Durbin, 1994; Knudsen & Hein, 2003) can then handle the alignment and they have proven to be very accurate in guessing the structure in that context. With a small family of poorly conserved RNA sequences, a good starting alignment can hardly be constructed, resulting usually in the inapplicability of the methods based on pre-alignment. In that second context it makes sense to seek the alignment and the structure at the same time. Sankoff (Sankoff, 1985) pioneered the field by exhibiting a set of recursions to optimally compute the best structural alignment of two RNA sequences when the structure is not known a priori. These recursions can be straightforwardly extended to N sequences and can also handle the energy parameters traditionally used for energy minimization (Mathews *et al.*, 1999). Although the time and space complexities for two sequences remains polynomial (resp. $O(n^6)$ and $O(n^4)$ if their lengths are the same order of magnitude n), the algorithm is not applicable without heuristic adaptations. From this more abstract point of view, RNA structural alignment becomes a challenging problem in terms of algorithmic issues. Diverse heuristic ideas were applied in able to turn the recursions usable. The first implementation of the Sankoff recursions was DYNALIGN (Mathews & Turner, 2002; Harmanci *et al.*, 2007) that reduced the complexity to $O(M^2n^2)$ in space and $O(M^3n^3)$ in time, where M is a (tunable) hard constant which bounds the shift allowed between the two sequences. Havgaard *et al.* (2005, 2007) (FOLDALIGN) investigated further by combining different other restrictions, namely:

- **alignment banding**: like in DYNALIGN the maximal shift between the sequences is bounded by a constant δ
- **structural banding**: the default behavior for the program is to perform a local alignment and the maximal size for a common motif is bounded by λ
- **multi-loop restriction**: the structure bifurcativity is limited in multi-loops

The resulting space and time complexities in FOLDALIGN become $0(n^2\lambda\delta)$ and $0(n^2\lambda^2\delta^2)$. The Vienna RNA Package also proposes an alternative attempt PMComp (Hofacker *et al.*, 2004) which is implemented as part of the RNAfold program and makes use of the McCaskill algorithm to compute the probabilities of base pairings for a single sequence (McCaskill, 1990). The authors of FOLDALIGN then revisited the idea by integrating their banding heuristic to PMComp (Torarinsson *et al.*, 2007).

*to whom correspondence should be addressed

Although these heuristics allow the application of the algorithm of Sankoff on natural RNA sequences, still remain some recalcitrant contexts where the method stays inapplicable. When the sequences under consideration show a large difference of length, poor conservation at the primary level or important structural variations, all Sankoff-based method either do not apply or – if they do – the banding heuristics prevent the algorithm from finding the correct structure, while the memory and time consumption explodes. One of the main drawbacks of this kind of heuristics is their global nature: they cannot take advantage of local similarities in the sequences. When performing a global structural alignment, the shift restriction δ should be at least the difference of length between the sequences. FOLDALIGN bypass the later difficulty by doing local alignment as a default behavior. In this paper we prove that the banding can be local, which results in a huge gain, both in memory and time consumption. We also propose a strategy to guaranty optimality in our framework even if we are using a heuristic. Our proposal could be improved if it were combined with other heuristics (for instance a branching restriction like the bifurcation constraint of FOLDALIGN) but with the loss of the guaranty for optimality. The strategy is presented in terms of constraints, we believe indeed that the constraint paradigm (already in use for other kind of methods, like CONSAN (Dowell & Eddy, 2006) or STEMLOC (Holmes, 2005)) is helpful and relevant to model the forthcoming improvements of our method. We observe that the recursions of Sankoff are a natural combination of the two kinds of recursions it is supposed to extend: sequence alignment and secondary structure prediction. Likewise, a structural alignment may be modeled as a subgraph of a combination of two graphs - an alignment graph and a folding graph. Imposing structural constraints on those graphs may lead to efficient heuristic design. The authors of FOLDALIGN already proposed a combination of constraints on each of these graphs, namely alignment banding and structural banding ; we propose a framework that allows more accurate constraints.

1 APPLYING CONSTRAINTS ON THE RECURSIONS OF SANKOFF

Alignments – An alignment of two sequences may be seen as a subgraph of the full bipartite graph where the nodes are the respective positions in each sequence. Such a subgraph (see Figure 1) is said to be an alignment if the following conditions (1) and (2) hold. Moreover, we impose the extra restriction of maximality (3)

(1) the arity of each node is at most one
(2) there is no crossing edge (we assume the nodes are placed in the sequence order)
(3) no edge can be added without breaking one of the two former requirements

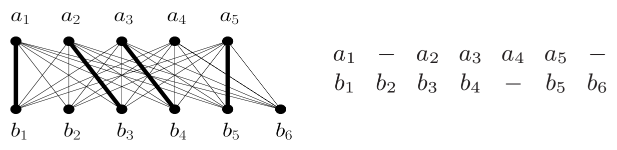
a_1 a_2 a_3 a_4 a_5
b_1 b_2 b_3 b_4 b_5 b_6
a_1 – a_2 a_3 a_4 a_5 –
b_1 b_2 b_3 b_4 – b_5 b_6

Fig. 1. An alignment can be seen as a subgraph of the full bipartite graph, we highlighted here the subgraph corresponding to a sample alignment

If the graph were not maximal in the sense of (3), it would mean that somewhere a deletion followed by an insertion would be preferred to a substitution. When the edges and remaining free bases are weighted according to the usual scoring schemes for two sequences, with linear or affine gap penalties (Needleman & Wunsch, 1970; Eppstein *et al.*, 1988, 1992*a*,*b*) such an alignment is automatically disqualified, since an appropriate shift of the corresponding bases would result immediately in an alignment of better score.

Secondary structure – Once again we consider the sequence as a list of nodes drawn in increasing order. Following the usual definition, a secondary structure is a subgraph of the full graph on these nodes with no crossing edge when drawn in a half plane (a so called outerplanar graph) and for which the node arities are at most one.

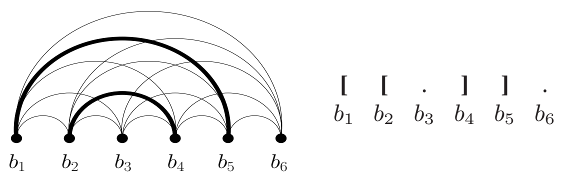
b_1 b_2 b_3 b_4 b_5 b_6
[[.]] .
b_1 b_2 b_3 b_4 b_5 b_6

Fig. 2. A secondary structure and the corresponding subgraph

In our framework, a structural alignment of two sequences can be represented as a combination of three graphs: an alignment graph and two structure graphs, like in Figure 3. In the following, we will call **structural envelope** any supergraph (built in the same fashion) that contains the structural alignment we seek.

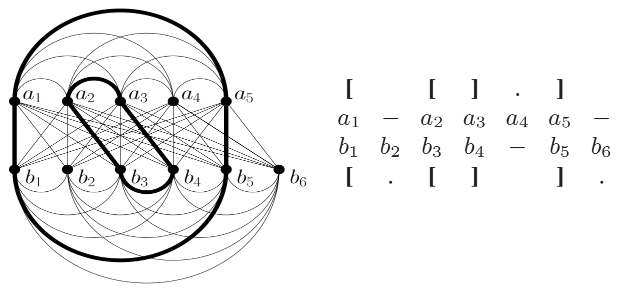
a_1 a_2 a_3 a_4 a_5
b_1 b_2 b_3 b_4 b_5 b_6
[[ ] .]
a_1 – a_2 a_3 a_4 a_5 –
b_1 b_2 b_3 b_4 – b_5 b_6
[. [ ]] .

Fig. 3. Example of a structural alignment seen as a subgraph of its structural envelope. The structural envelope holds no constraint here.

In practice, the structural envelope is not the full graph. For instance the steric constraints of the molecule imply that the minimal size for a loop should be three bases, consequently we can already remove all the edges that do not fulfill that constraint. In the next section, we investigate how the efficiency of the algorithm can be improved when extra constraints are imposed on the structural envelope.

The recursions of Sankoff – The recursion set used in RNA prediction methods based on free energy minimization (Jaeger *et al.*, 1989; Zuker *et al.*, 1999; Zuker, 2003; Hofacker, 2003) is a more complex version of the Nussinov formulas (Nussinov & Jacobson, 1980).

These formulas, in return, can be seen as such an energy minimization method in the oversimplified model where each pairing counts for one. Likewise, we express here the recursions given by Sankoff in the same simplified model and we discuss later how far we can extend it to a more sophisticated model with no loss in complexity.

We note $seq_0 = seq_0[1..n]$ and $seq_1 = seq_1[1..m]$ the two sequences in use. The Sankoff recursions in the simplified model of weighted pairing maximization are:

$$\mathbf{e}\,[i,j,k,l] := \mathbf{MAX}$$

$$\begin{cases} \bullet\, \mathbf{e}\,[i,j-1,k,l-1] + \mathbf{align\_base}(seq_0[j], seq_1[l]) \\ \bullet\, \mathbf{e}\,[i,j,k,l-1] + \mathbf{gap\_penalty}() \\ \bullet\, \mathbf{e}\,[i,j-1,k,l] + \mathbf{gap\_penalty}() \\ \bullet\, \mathbf{e}\,[i,x-1,k,y-1] + \mathbf{e}\,[x+1,j-1,y+1,l-1] + \\ \quad \mathbf{match\_pair}(seq_0[x], seq_0[j], seq_1[y], seq_1[l]) \\ \qquad [i \leqslant x \leqslant j] \qquad [k \leqslant y \leqslant l] \end{cases}$$

The value $\mathbf{e}\,[1,n,1,m]$ gives the best score for a structural alignment in that model. Any of the corresponding best alignments can be retrieved in linear time by tracing back into the dynamic programming matrix if the corresponding pairings that maximize the score were stored during the search, which would require only $0(n^2)$ space. In that form, the Sankoff recursions appear to be a combination of the recursions for sequence alignment (Smith & Waterman, 1981) and the recursions for secondary structure prediction of Nussinov. The algorithm can integrate any Boolean restraints imposed (by the user or by automated methods) on the structural envelope by simply assigning an infinite score to the forbidden matches. What we show is that these constraints can be used to reduce both memory and time consumption.

Constraining the alignment – In the following, any pair of objects (bases, indices, segments...) are called **foreign** if they do not belong to the same sequence. Two foreign bases (or their index in the sequence) are said to be **alignables** if they are allowed to participate to the final alignment. The resulting subgraph is called the **alignment envelope**. In Figure 4, we show such an alignment envelope where we only represented the edges for two foreign indices. For a given index i of arity M_i in the first sequence, we note $a_i = a_i[0 \,..\, M_i] = \{\ a_i[0]\ ,\ ..\ ,\ a_i[M_i]\ \}$ the list of its alignable bases, and for k in the second sequence, of arity N_k, the list of its alignable bases in the first sequence is noted b_k.

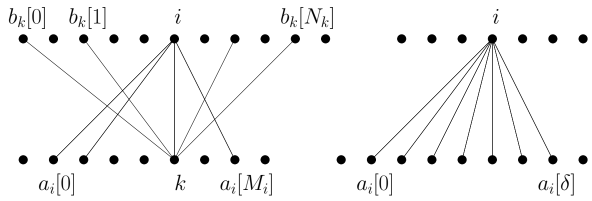

Fig. 4. On the left are represented the lists a_i and b_k of alignable bases for some indices i and k. We note M_i the length of a_i and N_k the length of b_k. The figure on the right shows the list of contiguous indices for an arbitrary index i when using a sequence banding heuristic with a fixed global constant δ, like in DYNALIGN and FOLDALIGN.

Alignable bases in a list do not need to be contiguous in the sequence. The sequence banding heuristic used in DYNALIGN and FOLDALIGN would give for any index a list of contiguous bases of fixed length δ where the value for δ is a global constant.

Our claim is that we just need to allocate for each pair (i,j) a square matrix of size $M_i \times M_j$ (Figure 5). The value in the cell $[i,j,x,y]$ stores the score for the best structural alignment between the segments $seq_0[i..j]$ and $seq_1[a_i[x]..a_i[y]]$. When looking a bit closely to the recursion formulas, it appears that during the computation process, we may fall out of the allocated part. We prove that the best structural alignment between any segments $seq_0[i..j]$ and $seq_1[k..l]$ can be retrieved in constant time from the allocated part.

Figure 5 displays a possible layout for the 4-dimensional matrix, where the outer matrix is related to the first sequence and the smaller inner matrices to the second one. But the modeling is actually perfectly symmetric. Likewise, if we had focused on the second sequence and allocated the inner matrices based on the lists b_k and b_l, the overall matrix would result in exactly the same size. The size of this 4-dimensional matrix $\sum_{(i,j)} M_i \times M_j = (\sum_i M_i)^2 = (\sum_k N_k)^2 = \sum_{(k,l)} N_k \times N_l$ is the square product of the number of edges in the alignment envelope, and the matrix itself can be considered as labeled by the edges of the alignment envelope: each 4d-cell $[i,j,x,y]$ is indeed related to the unique pair of edges $(i, a_i[x])$ and $(j, a_j[y])$.

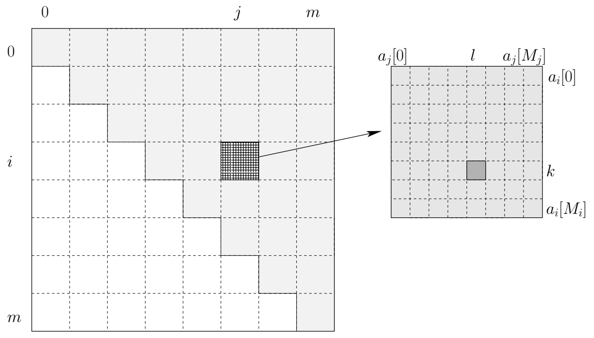

Fig. 5. The 4-dimensional matrix used for the computation of the best structural alignment can be represented as a bi-dimensional matrix, each cell being a bi-dimensional matrix. The value in the cell $[i,j,x,y]$ stores the score for the best structural alignment between the segments $seq_0[i..j]$ and $seq_1[a_i[x]..a_i[y]]$.

Claim – The score $\mathbf{e}\,[i,j,k,l]$ can be retrieved in constant time from the reduced matrix of Figure 5.

Proof – Let's assume that during the computation $\mathbf{e}\,[i,j,k,l]$ falls out of the allocated matrix. This means that either i and k are not alignable and/or j and l are not alignable. We detail the right hand case, the other one is symmetric. Given two non-empty foreign segments $[i..j]$ and $[k..l]$, if j and l are not alignable then either j align with some base in $[k..l]$, or l with some base in $[i..j]$. Otherwise j and l would both create an indel and the optimal alignment of the two segments would then result in a double indel which was previously assumed to be more expensive than a substitution. Let's assume first that l aligns with a base in $[i..j]$. Consequently $[i..j] \bigcap b_l \neq \emptyset$. Let's call $\beta_l = max([i..j] \bigcap b_l)$. Every base of the segment $[\beta_{l+1}..j]$ has no alignable partner in $[k..l]$ or it would contradict the maximality of β_l. The remaining sequence $seq_0[\beta_{l+1}..j]$ has no other possibility than being deleted, resulting in a $(j - \beta_l)$ long gap in the alignment.

This gives:

$$\mathbf{e}\,[i,j,k,l] = \mathbf{e}\,[i,\beta_l,k,l] + (j-\beta_l) * \mathbf{gap\_penalty}$$

If l does not align, then j must align with some base in $[k..l]$ and we pose $\alpha_j = max([k..l] \bigcap a_j)$. In that case we must have:

$$\mathbf{e}\,[i,j,k,l] = \mathbf{e}\,[i,j,k,\alpha_j] + (l-\alpha_j) * \mathbf{gap\_penalty}$$

Note that α_j and β_l depend only on three indices and can be precomputed in cubic time so that the former retrieval can then be managed in constant time when the routines are calling an index out of the matrix. The left hand case with indices i and k is symmetric. As the indices are not dependent, a single test on each of the four indices is enough to have the certitude to fall back in the allocated part.

Discussion – The demonstration implicitly makes two important assumptions that are worth emphasizing: (1) we use linear gap penalty and (2) we assume that a double indel is always worse than a substitution. The proof may not be valid for more sophisticated scoring schemes and would also have to be adapted for more than two sequences, where the second assumption is not always true in an optimal multiple alignment. This points remain to be investigated. However, from a theoretical point of view, we believe that the improvements are meaningful enough to be exposed and, from a more practical point of view, the next section will demonstrate that even with a simple model, either for sequence alignment and/or for structure prediction, the important computational gain allows to use the algorithm with success on difficult data sets that usually remain inaccessible to other Sankoff-based methods. In the next section, we expose a strategy to choose the constraints, we exemplify on natural sequences and also compare our results to a similar Sankoff-based algorithm.

2 STRATEGY AND EXPERIMENTAL RESULTS

The algorithm we propose was implemented using a three-fold scheme which can be summarized like this: seeking the best structural alignment among a set of suboptimal sequence alignments. Our hypothesis is that the best structural alignment should not be so far from the best (non-structural) sequence alignment. Therefore it makes sense to believe that the optimal alignment we seek is indeed a subgraph of the alignment envelope, when the later was set as a graph of suboptimal sequence alignments.

2.1 Method

We provide and discuss an implementation based on a reduced energy model that partially takes into account the stabilizing effect of base pair stacking in stems. Our scoring scheme for sequence alignment is rather empirical, too. The present implementation – named **ARNICA** – was specially aimed at exemplifying the gain both in space and time consumption when running on live sequences. We give clear evidence that, even in that simplified form, ARNICA shows a remarkable tendency to remain very stable in terms of performance, whatever the characteristics of the data and proves to be competitive with other methods (we only compared to FOLDALIGN for simplicity, as it is one of the prominent Sankoff-based structural alignment method).

We proceed in three steps:

- **1. Setting alignment constraints** - we build the graph of alignable bases by computing all the alignments within a user-specified distance of the optimum alignment value
- **2. Setting structural constraints** - we compute all the pairing probabilities for each sequence and filter with a threshold
- **3. Seeking common folding and alignment** - we compute the optimal structural alignment with the recursions of Sankoff

1. Setting alignment constraints – During the first phase, we use a variant of the standard alignment with affine gap penalty (open_cost = -80, elongation_cost = -30) reminiscent of the algorithm of Waterman (Watermann, 1983) that gives for each pair of positions (i,j) the score of the best alignment when the bases at position i in the first sequence and j in the second one are imposed to be aligned. The variant of the algorithm has the same algorithmic complexities than for standard alignment methods: basically the score for the best alignment when the foreign bases i and j are imposed to be aligned is the sum of the score for the best prefixes alignment (*ie* the best alignment for $seq_0[1..i-1]$ and $seq_1[1..j-1]$) and for the best suffixes alignment (the best alignment of $seq_0[i+1..n]$ and $seq_1[j+1..m]$), plus the cost for the substitution of $seq_0[i]$ and $seq_1[j]$. The exact algorithm is just a bit more sophisticated when affine gap penalty is in use but the complexity remains unchanged.

Then we build the (Boolean) adjacency matrix of the alignment envelope by simply applying a threshold (thd). A pair i,j will be allowed to align if there exists at least one alignment passing by this coordinate for which the difference of scores with the best alignment does not exceed thd.

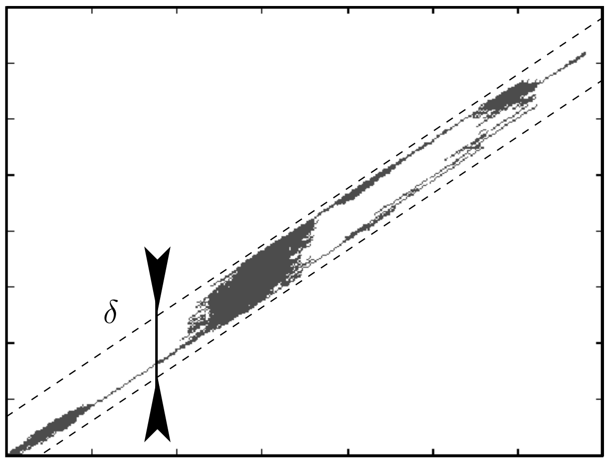

Fig. 6. Suboptimal alignments of two RNase P RNA (D.desulfuricans vs A.eutrophus) showing alternative alignment paths resulting from large zones of deletion. To reach any of these suboptimal alignments with a banding heuristic, the value chosen for the allowed shift δ has to encompass all the possible paths.

The size of the 4-dimensional matrix to be allocated in the last step obviously increases with the threshold. If the threshold is chosen to be infinite, then the alignment envelope is the full bipartite graph and there is no gain over the complexities of the Sankoff recursions. In practice however, our framework is quite effective and far more efficient than any alignment banding heuristic, which is no more than a peculiar case of it. The algorithmic complexities for a Sankoff-based pairwise secondary structure alignment with alignment banding δ are $O(\delta^2 n^2)$ in space and $O(\delta^3 n^3)$ in time (we suppose that $m \sim n$). They can be reformulated as $O(\alpha^2)$ and $O(\alpha^3)$, where $\alpha \sim \delta n$ is the size of the diagonal « band » corresponding to the alignment envelope induced by the banding heuristic. More precisely, α is the size of the *true* zone in the adjacency matrix of the alignment envelope. As shown on Figure 6, this zone is an exact diagonal band in the banding heuristic, whereas it can be a tunable zone in our framework. The sequences chosen for this example show important variations in structure, resulting in large zones of insertion / deletion at the primary level. The adjacency matrix of the alignment envelope shows alternative paths for the sought alignment and the use of alignment banding would imply a large value for δ to be able to reach the same possible alignments.

2. Setting structural constraints – In the second phase, the pairing probabilities are computed for each sequence with the McCaskill algorithm (McCaskill, 1990) and the graph of possible pairings is filtered: two bases for which the pairing probability is less than 1% are not allowed to pair. This has no consequence on the computation space/time but simply increases the quality of the solutions found by discarding spurious base pairs.

3. Seeking common folding and alignment – Then we compute the optimal folding and alignment with the recursions of Sankoff. We use a linear gap penalty scheme for the alignment part of the score and a probability-based score for the structural part. The combination of two scores of different nature (alignment score, structure probability) is a difficult issue and a full problem in itself, related to the « paradox » of structural alignment, that is not discussed in the present paper, we simply precise that the optimality is relative to our model. The model we use take into account only indirectly the stabilizing and destabilizing effects of stacking. Integrating the full thermodynamic model would provide more biologically accurate results but also call for non-straightforward developments and adaptations. The next section demonstrate that the method already appears competitive despite the simplicity of the model and that it clearly outperforms other programs based on the same recursions when the data presents uneasy features.

2.2 Experiments

In the following we settled some experiments in which we compare our results to FOLDALIGN. The program FOLDALIGN does not allocate the whole needed memory at once: in the comparisons, we always display the maximum amount of memory in use by the program during the computation. To keep in reasonable space and time limits, and given that a memory allocation of several hundreds of Mb would usually result in hours of computation, we stopped the computation whenever the estimated needed resources for memory were above 300Mb. All the tests were run on an IBM thinkpad T40 (pentium 1.5GHz - RAM 512Mb). For our program ARNICA, the threshold parameter (thd) was tuned to different increasing values. We selected two families of sequences - tRNA and RNase P RNA. The cloverleaf structure of tRNA is known from a long time and the available alignments can be considered very reliable. This first data set is aimed at testing and demonstrating the limits of the model we are using. The second family of sequences shows deep variations in structure, which make them difficult candidates for all Sankoff-based methods, as mentioned by (Gardner & Giegerich, 2004).

tRNA – We selected the 20 first tRNA of the seed alignment RF00005 of the RFAM database (Griffiths-Jones *et al.*, 2003) from which we discarded 4 sequences that were too close in order to have a maximum pairwise sequence identity of 80% (the average on this set is actually 59%). Each time, the predictions are compared with the known structure: Table 1 displays the performances of ARNICA and FOLDALIGN on this sample set. Times are in seconds, memory usage in Mb, specificity (spec) and sensitivity (sens) are given by the formulas:

$$spec = \frac{number\ of\ true\ predicted\ pairings}{total\ number\ of\ predicted\ pairings}$$

$$sens = \frac{number\ of\ true\ predicted\ pairings}{number\ of\ pairings\ in\ the\ known\ structure}$$

The default feature of FOLDALIGN is to compute a local structural alignment with a banding value $\delta = 25$ but it is possible to ask for a global alignment if the difference in length for the sequences does not exceed 25. We display both results.

	option	spec	sens	time	space
FOLDALIGN	local	87.1%	72.0%	3.6	6.6
	global	86.6%	89.0%	4.5	9.6
ARNICA	thd 0	79.7%	66.8%	0.2	< 0.1
	thd 10	79.5%	67.9%	0.2	< 0.1
	thd 30	77.8%	69.3%	0.3	< 0.1
	thd 50	76.7%	71.2%	0.3	< 0.1
	thd 100	72.7%	72.6%	0.6	0.8

Table 1. Average performance of ARNICA and FOLDALIGN on a set of tRNA [specificity (spec) - sensitivity (sens) - time in seconds - space in Mb]

When the distance threshold to the optimal alignment used in the first phase is increased, the sensibility of ARNICA is increased too. In the meantime, the specificity shows a decrease. The average specificity (around 75% on this example) which is less than the 87% of FOLDALIGN is related to the limits of the model we use. When enlarging the threshold, the algorithm has more flexibility for the choice of pairings and, as the stabilizing effect of stems is not taken into account in the recursions, the program often chooses pairs that have higher score, regardless of their possible stacking with neighbors. The score provided by the algorithm of McCaskill favour pairings that *often stack in suboptimal structures* but this is not enough to always drive ARNICA toward a preference for stacked pairs. A closer look at the predicted structures reveals that this is indeed what happens and remind us the limit of the model in use. In compensation, we gain on this set a factor 10 both in computation time and space requirements. On the next sample set, we focus on a more difficult case where the performances of ARNICA remain more or less the same in terms of specificity, overtaking the other methods.

RNase P – In this second sample test, we ran the two programs on 7 RNase P RNA of the alpha subdivision taken in the database of Brown (Brown, 1999). These sequences are around 400 bases long with an average identity rate of 64%. They have the particularity to show large differences in length, coming from important variations in the shared structure. We did not compute the cofolding when the estimated size of the matrix was to exceed 300Mb. Table 2 gives the average performances of ARNICA with different values for the threshold, compared to FOLDALIGN ; the results for a medium threshold of 100 are detailed in Table 3. For FOLDALIGN, we use the default option (local alignment), as the global option can seldom be chosen, the difference of length being too important between the sequences. However, we indicate in Table 3 the percentage of sequence covered by the structural alignment predicted. When this coverage is close to 100%, the difference between local and global is weak and comparing the performances is meaningful.

	option	spec	sens	time	space
FOLDALIGN	local	56.2%	40.5%	1107	142.1
ARNICA	thd 0	73.6%	43.3%	6	16.5
	thd 10	74.1%	45.9%	7	16.7
	thd 30	75.7%	51.0%	10	17.5
	thd 50	76.2%	55.0%	20	19.1
	thd 80	75.7%	58.3%	77	23.7
	thd 100	74.7%	58.7%	148	29.0
	thd 150	73.9%	60.5%	536	46.5
	thd 200	73.2%	61.0%	1079	64.4

Table 2. Average performance of ARNICA and FOLDALIGN on a set of RNase P (alpha subdivision) [specificity (spec) - sensitivity (sens) - time in seconds - space in Mb]

On this data set, FOLDALIGN demonstrates a loss of impetus, due to the restrictions imposed by the heuristics, whereas ARNICA performs better at any point of view (specificity, sensitivity, time and memory usage). Like in the previous experiment with tRNA, ARNICA remains fast and low memory consuming. Here the average gain in computational time and memory with a threshold 100 is by a factor 7 and 5, and the correctness of ARNICA is numerically close (the specificity is neighboring 75% for each data set). This globally stable behavior, whereas the features and difficulty of the data are quite different, is a promising advantage for the integration of a more complete model.

3 CONCLUSIONS

In this paper we proposed a refinement of the heuristics commonly used by the Sankoff-based pairwise secondary structure RNA prediction methods. We exposed a strategy based on the constraint paradigm to extend the possibilities for heuristic design. We also came back to a more simple model that constitutes the core of these methods, for which we proved that our framework is valid. In our ongoing developments, we aim at incorporating a more complete thermodynamic model and refining even further the method by allowing dynamic restraints on the graphs. The stability of ARNICA over the variability of data is a major asset for our ongoing improvements.

REFERENCES

Brown, J.W.. (1999) The Ribonuclease P database. *NAR*, **27** (314). `http://www.mbio.ncsu.edu/RNaseP/`.

Dowell, R. D. & Eddy, S. R. (2006) Efficient pairwise rna structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, **7**, 400+.

Eddy, S.R.. & Durbin, R. (1994) RNA sequence analysis using covariance models. *NAR*, **22**, 2079–2088.

Eppstein, D., Galil, Z. & Giancarlo, R. (1988) Speeding up dynamic programming. In *Proceedings of the29th IEEE Annual Symposium on Foundations of Computer Science* pp. 488–496 IEEE Computer Society Press, White Plains, NY.

Eppstein, D., Galil, Z., Giancarlo, R. & Italiano, G. F. (1992*a*) Sparse dynamic programming ii: convex and concave cost functions. *J. ACM*, **39** (3), 546–567.

Eppstein, D., Galil, Z., Giancarlo, R. & Italiano, G. F. (1992*b*) Sparse dynamic programming i: linear cost functions. *J. ACM*, **39** (3), 519–545.

Gardner, P. P. & Giegerich, R. (2004) A comprehensive comparison of comparative rna structure prediction approaches. *BMC Bioinformatics*, **5** (1).

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A. & Eddy, S.R.. (2003) RFAM: an RNA family database. *NAR*, **31** (1), 439–441.

Harmanci, A. O., Sharma, G. & Mathews, D. H. (2007) Efficient pairwise rna structure prediction using probabilistic alignment constraints in dynalign. *BMC Bioinformatics*, **8**, 130+.

Havgaard, J. H., Lyngs ● , R. B., Stormo, G. D. & Gorodkin, J. (2005) Pairwise local structural alignment of rna sequences with sequence similarity less than 40%. *Bioinformatics*, **21** (9), 1815–1824.

Havgaard, J. H., Torarinsson, E. & Gorodkin, J. (2007) Fast pairwise structural rna alignments by pruning of the dynamical programming matrix. *PLoS Computational Biology*, **3** (10), e193+.

Hofacker, I. L. (2003) Vienna rna secondary structure server. *Nucleic Acids Res*, **31** (13), 3429–3431.

Hofacker, I. L., Bernhart, S. H. & Stadler, P. F. (2004) Alignment of rna base pairing probability matrices. *Bioinformatics*, **20** (14), 2222–2227.

Holmes, I. (2005) Accelerated probabilistic inference of rna structure evolution. *BMC Bioinformatics*, **6** (1).

Jaeger, J.A.., Turner, D.H.. & Zuker, M. (1989) Improved predictions of secondary structures for RNA. *PNAS*, **86**, 7706–7710.

Knudsen, B. & Hein, J. (2003) Pfold: rna secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res*, **31** (13), 3423–3428.

Marti-Renom, Marc, A., Capriotti & Emidio (2008) Computational rna structure prediction. *Current Bioinformatics*, **3** (1), 32–45.

Mathews, D.H.., Sabina, J., Zuker, M. & Turner, D.H.. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *JMB*, **288**, 911–940.

Mathews, D.H.. & Turner, D.H.. (2002) Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *JMB*, **-**, in press.

McCaskill, J.S.. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

Needleman, S. B. & Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, **48** (3), 443–453.

Nussinov, R. & Jacobson, A.B.. (1980) Fast algorithm for predicting the secondary structure of single stranded RNA. *PNAS*, **77**, 6309–6313.

O'Brien, E.A.. & Higgins, D.G.. (1998) Empirical estimation of the reliability of ribosomal RNA alignments. *Bioinformatics*, **14** (10), 830–838.

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

Shapiro, B. A., Yingling, Y. G., Kasprzak, W. & Bindewald, E. (2007) Bridging the gap in rna structure prediction. *Current Opinion in Structural Biology*, **17** (2), 157–165.

Smith, T.F.. & Waterman, M.S.. (1981) Identification of common molecular subsequences. *JMB*, **147**, 195–197.

Torarinsson, E., Havgaard, J. H. H. & Gorodkin, J. (2007) Multiple structural alignment and clustering of rna sequences. *Bioinformatics*, .

Watermann, M. S. (1983) Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Applied Mathematical Sciences*, **80**, 3123–3124.

Zuker, M. (2003) MFOLD web server for nucleic acid folding and hybridization prediction. *NAR*, **31** (13), 1–10.

Zuker, M., Mathews, d. & Turner, d. (1999) *Algorithms and Thermodynamics for RNA Secondary Structure Prediction: A Practical Guide.* NATO ASI Series, Kluwer Academic Publishers.

				ARNICA						FOLDALIGN						
sequence 1	sequence 2	id	δl	**spec1**	**sens1**	**spec2**	**sens2**	**time**	**mem**	spec1	sens1	spec2	sens2	time	mem	cov
C.crescentus	A.tumefaciens	68%	4	**70%**	**69%**	**74%**	**73%**	**46**	**22.7**	69%	66%	65%	62%	1428	170.5	99%
R.capsulatus	A.tumefaciens	67%	3	**83%**	**81%**	**82%**	**84%**	**31**	**21.4**	59%	54%	74%	71%	1647	181.4	99%
R.capsulatus	C.crescentus	67%	7	**77%**	**58%**	**73%**	**58%**	**55**	**23.6**	49%	33%	45%	32%	1090	169.9	71%
R.palustris	A.tumefaciens	75%	77	**77%**	**59%**	**77%**	**78%**	**87**	**30.1**	36%	17%	42%	26%	1329	200.6	58%
R.palustris	C.crescentus	65%	81	**83%**	**57%**	**77%**	**71%**	**114**	**32.1**	19%	12%	27%	23%	1840	226.5	87%
R.palustris	R.capsulatus	67%	74	**82%**	**61%**	**78%**	**74%**	**287**	**42.4**	24%	16%	27%	23%	2326	262.7	91%
R.prowazekii	A.tumefaciens	60%	17	**75%**	**50%**	**74%**	**48%**	**37**	**21.3**	70%	59%	66%	53%	516	79.0	87%
R.prowazekii	C.crescentus	56%	13	**84%**	**52%**	**80%**	**49%**	**66**	**24.1**	65%	52%	61%	48%	423	76.1	84%
R.prowazekii	R.capsulatus	56%	20	**80%**	**47%**	**83%**	**45%**	**39**	**21.5**	61%	51%	61%	48%	585	75.4	86%
R.prowazekii	R.palustris	60%	94	**78%**	**53%**	**81%**	**41%**	**161**	**31.6**	35%	28%	40%	24%	554	83.9	76%
R.rubrum	A.tumefaciens	74%	27	**70%**	**57%**	**71%**	**65%**	**43**	**23.2**	46%	43%	62%	65%	1991	208.6	98%
R.rubrum	C.crescentus	69%	31	**60%**	**51%**	**64%**	**61%**	**28**	**21.3**	62%	57%	63%	64%	2030	214.4	97%
R.rubrum	R.capsulatus	68%	24	**77%**	**63%**	**81%**	**71%**	**46**	**24.0**	68%	63%	75%	74%	2253	252.2	96%
R.rubrum	R.palustris	74%	50	**80%**	**68%**	**83%**	**60%**	**167**	**34.5**	34%	30%	43%	32%	2960	299.5	91%
R.rubrum	R.prowazekii	55%	44	**39%**	**26%**	**59%**	**45%**	**728**	**57.7**	66%	47%	77%	64%	588	94.4	80%
Wolbachia-sp	A.tumefaciens	60%	54	**67%**	**65%**	**72%**	**59%**	**71**	**23.2**	69%	35%	69%	29%	277	65.8	51%
Wolbachia-sp	C.crescentus	56%	50	**72%**	**49%**	**76%**	**44%**	**95**	**25.3**	56%	36%	61%	33%	309	66.0	64%
Wolbachia-sp	R.capsulatus	59%	57	**77%**	**73%**	**78%**	**60%**	**87**	**24.8**	81%	39%	81%	31%	265	62.9	48%
Wolbachia-sp	R.palustris	58%	131	**81%**	**78%**	**85%**	**53%**	**744**	**54.5**	33%	10%	57%	11%	274	69.2	28%
Wolbachia-sp	R.prowazekii	67%	37	**61%**	**42%**	**70%**	**42%**	**77**	**23.3**	70%	39%	79%	34%	195	55.0	58%
Wolbachia-sp	R.rubrum	60%	81	**79%**	**76%**	**68%**	**49%**	**97**	**25.6**	62%	42%	51%	26%	376	79.0	61%
average				**74%**	**59%**	**76%**	**59%**	**148**	**29.0**	54%	39%	58%	42%	1107	142.1	

Table 3. Performance of ARNICA and FOLDALIGN on a set of RNase P (alpha subdivision). Each sequence is folded together with each other. We display their percentage of identity and their difference in length (their average length is around 400 bases) [spec1 and spec2 (specificity for each of the two sequences) - sens1 and sens2 (sensitivity) - time is in seconds - mem is in Mb - cov is the percentage of sequence covered by the local alignment given by FOLDALIGN (ARNICA is global)]