PLANAR: RNA Sequence Alignment using Non-Affine Gap Penalty and Secondary Structure

O. GILL *
Courant Inst., NYU;
E-mail: gill@cs.nyu.edu.

N. RAMAKRISHNAN
Virginia Tech.;
Email: naren@cs.vt.edu

B. MISHRA
Courant Inst., NYU.;
Email: mishra@nyu.edu

Abstract

An important component of bioinformatics research is aimed at finding evolutionary relationships among species, since it allows us to better understand various important biological functions as they emerged in these species. These tools simultaneously trace the associated evolutionary history. In this context, sequence alignment is commonly used to understand similarities among the species by comparing their genomic, transcriptomic and proteomic sequence data. In our earlier research, we devised PLAINS (Piecewise-Linear Alignment with Important Nucleotide Seeker) to align genomic DNA sequences using a general class of gap distribution model. This paper proposes PLANAR (Piecewise-Linear Alignment for Nucleotides Arranged as RNA), a variant of PLAINS designed for aligning RNA sequences, while properly accounting for their secondary structures. This paper presents an overview of the PLANAR algorithm, and compares it to other competing RNA alignment tools, while emphasizing many interesting correlations discovered in the process.

Introduction

Within an organism, a small percentage of its genome transcribes into RNA. Even a smaller percentage of that RNA, namely just the mRNA, is translated into proteins. The transcription occurs in the nucleus cued by transcriptional factors and modulated by enhancers and repressors. The translation occurs in the cytoplasm, and involves ribosomes, rRNA, tRNA, and other complexes.

Until recently, it was believed that the cellular functions were primarily carried out by the protein coding genes, and most cellular functions found in noncoding RNA (such as rRNA and tRNA) were homologous to coding regions. It has now been argued that this assumption is most likely incorrect, because it fails to explain how, despite its functional complexity, there could be as few as just 25 thousand protein-coding genes in the human genome, which is only twice as many genes as in Drosophila (although the human genome is far longer than just twice the length of Drosophila). A simple resolution to this dilemma is offered by the suggestion that many more complex cellular functions in humans and other mammals must be carried out by noncoding RNA.

The discovery of miRNA's has been regarded as a scientific breakthrough, as they suggest a new and a large class of eukaryotic regulatory elements for genes. Evidence for its important functional roles has been concluded from various phenotypic, expressional, and evolutionary analyses, although the number of miRNA's with known functions remain still fairly miniscule. Understanding all of its pertinent features requires powerful in-vivo/vitro and in-silico methods, since such analyses provide a characterization of regulators and their targets. A few known miRNA functions have been characterized in leaf development, timing of larval transitions, apoptosis, fat metabolism, and insulin secretion.

The discovery of miRNA's has not only led to further identification of other noncoding RNA, but also many new bioinformatic research problems. As an example, note that the bioinformatics of identifying correlations among miRNA's differs significantly from that of DNA or proteins because, unlike with DNA or proteins, the sequence similarities are not sufficient to identify the functional correlations among these short RNA sequences. For RNA, its functionality is also tied to its secondary structure, and is affected by the fact that its weaker nucleotide bonds (e.g., relative to dsDNA) endow it with far more flexibility in its structure. Consequently, in order to effectively align RNA sequences, we must account for secondary structures in addition to the sequences themselves.

The secondary structure of RNA features many loops (both internal and external hairpin loops), multi-loops, bulges, and pseudoknots. See Figure 1 for details. The hairpins help to stabilize the formation of complexes for coworking elements, e.g., proteins. In order to reduce the runtime of our alignment algorithms and for the sake of simplicity, we will ignore pseudoknots, and refer the readers to the work of Eddy and Rivas (Rivas & Eddy 1999) for a more detailed discussion of alignment using pseudoknots.

*Work supported by NSF ITR, Internal NYU Grants, and Army PCRP.

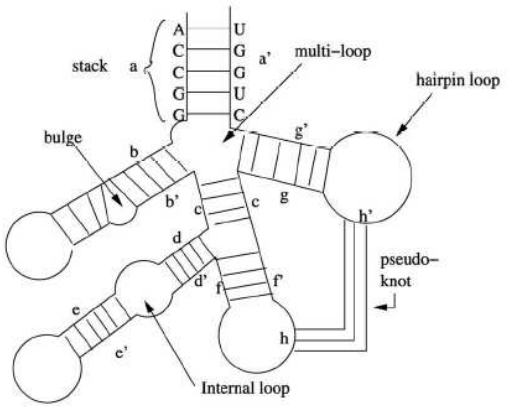

Figure 1: An example of a secondary structure of an RNA with bulges, hairpin loops, internal loops, multi-loops, and pseudoknots.

Structural Alignment

Often in this paper, we will formulate our problems as primarily aligning X against Y[1] where we only concern ourselves with X's secondary structure, disregarding Y's structure. We will adjust the alignment subsequently using Y's secondary structure as needed. This approach is unlike Liu et al. (Liu *et al.* 2005), who align X and Y using both X and Y's secondary structures simultaneously to generate the alignment. We have chosen not to use such a method in order to avoid being excessively reliant on secondary structures. PLANAR further differs from its derived RNA alignment tools in that it uses piecewise-linear gap functions—not a linear gap formula without a penalty for opening a gap that is typical to RNA alignment.

Binarization

Binarization is the process of converting an RNA sequence X into a tree T_X using its secondary structure. This tree contains nodes of labels 'P', 'L', 'R', 'B' or 'E' corresponding respectively to paired positions, left-only positions, right-only positions, bifurcations, and end-points. Nodes of 'B' label have two children; nodes of 'E' label are leaves; and all other nodes have exactly one child. Constructing T_X from X with the secondary structure in this manner is similar to the Binarization mentioned in the CMSAA algorithm of Eddy et al. (Eddy 2002). Figure 2 elaborates further. Note that if $|X| = m$, then T_X has at most $2m$ nodes.

PLANAR Alignment Formulation

Given two sequences X and Y with tree T_X constructed for X, let $V(v, i, j)$, denote the alignment for v's subtree aligned against $Y[i : j]$, where v is a node from tree T_X. Also, let $|v|$ denote the number of nodes in v's subtree, including v itself. Next, let $LCB(u, v)$ denote a Boolean formula that holds only if, for nodes u and v from T_X, u is

[1] with $|X| = m$, $|Y| = n$

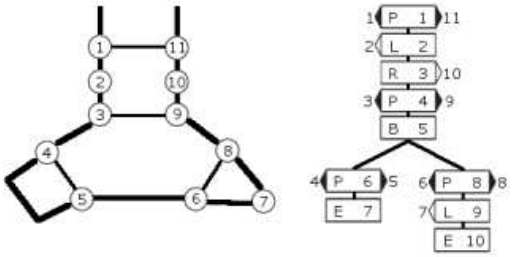

Figure 2: In the drawing to the left, we see a sample secondary structure for an RNA sequence X with indices numbered 1 thru 11, and bonds between node pairs $(1, 11)$, $(3, 9)$, $(4, 5)$, and $(6, 8)$. On the right, we see how CMSAA would binarize X into tree T_X using the appropriate 'P', 'L', 'R', 'B' and 'E' nodes.

below v, and it is possible to traverse node-by-node from v to u without encountering a bifurcation node (node of 'B' label).

To assist in computing table V, we use tables G, D, E, and F, where G denotes the alignment ending at a matched or mismatched position, F denotes the alignment ending with X (or T_X in this case) aligned against a gap, D denotes the alignment ending with the left portion of Y aligned against a gap, and E denotes the alignment ending with the right portion of Y aligned against a gap. Furthermore, we will also make use of a p-part piecewise-linear gap function $ww(\cdot)$, as well as $s(\cdot, \cdot)$ and $b(\cdot, \cdot, \cdot, \cdot)$ for unbound and bound position scores for matches and mismatches. All of this notation is similar to that used in (Gill, Zhou, & Mishra 2005) for the PLAINS formula for DNA alignment, except that we are also accounting for matters related to secondary structure here. $V(v, i, j)$ is computed as follows:

- Base Cases:
  - If v's label is 'E', then:
$$V(v, i, j) = ww(j - i + 1)$$
  - If $i > j$, then:
$$V(v, i, j) = ww(|v|)$$
- Recursive Cases:
  - If v's label is 'B', then:
$$V(v, i, j) = \max_{i-1 \le k \le j} [V(v.left, i, k) + V(v.right, k + 1, j)]$$
  - If v's label is NOT 'B', then:
$$\begin{aligned} V(v, i, j) &= \max\{D(v, i, j), E(v, i, j), F(v, i, j), G(v, i, j)\} \\ D(v, i, j) &= \max_{i+1 \le k \le j+1} [V(v, k, j) - ww(k - i)] \\ E(v, i, j) &= \max_{i-1 \le k \le j-1} [V(v, i, k) - ww(j - k)] \\ F(v, i, j) &= \max_{u \text{ s.t. } LCB(u,v)} [V(u, i, j) - ww(|v| - |u|)] \end{aligned}$$
  - If v's label is 'L', then:
$$G(v, i, j) = s(X[l_v], Y[i]) + V(v.child, i + 1, j)$$
  - If v's label is 'R', then:
$$G(v, i, j) = s(X[r_v], Y[j]) + V(v.child, i, j - 1)$$

– If v's label is 'P', and $i < j$ then:

$$G(v,i,j) = b(X[l_v], X[r_v], Y[i], Y[j]) + V(v.child, i+1, j-1)$$

– Otherwise:

$$G(v,i,j) = -\infty$$

We traceback from $V(root, 1, n)$ to get the alignment for all of T_X aligned to $Y[1:n]$. Note that the 'P' nodes of T_X facilitate more meaningful alignments based on the secondary structure. Also, for algorithmic simplicity and reliance on secondary structure, our formulation for gap penalties treats each linear chain in T_X independently of each other.

Using an approach similar to Needleman-Wunsch implies that our runtime is $O((m+n)n^2m_1+n^3m_2) = O(m^2n^2 + n^3m)$ and the space usage is $O(n^2m)$, where m_1 is the number of nodes with zero or one child, and m_2 is the number of nodes with two children.

PLANAR Linked-List Assistance and Space Reduction

However, we improve the space complexity by computing the V tables differently than Needleman-Wunsch. By using Linked-List Assistance and CMSAA's space reduction strategies, PLANAR improves both the runtime and memory usages.

In Miller-Myers(Miller & Myers 1988), and subsequently in the PLAINS paper(Gill, Zhou, & Mishra 2005), the Miller-Myers Linked-List Assistance has been advantageously applied to reduce the runtime of the Needleman-Wunsch algorithm. Using a similar tactic for the PLANAR alignment formula, when v is fixed, we treat each $n \times n$ deck of form $V(v,\cdot,\cdot)$ separately (and similarly for D, E, F, and G) and save the t most recently computed decks, where t is treated as a constant. For each row i, we use a list LD_i to reduce the lookups in computing $D(v,i,\cdot)$. For each column j, we use a list LE_j to reduce the lookups in computing $E(v,\cdot,j)$. We finish computing each deck before moving on to the next one, and we can empty and reuse each list LD_i and LE_j when changing our v value. For each i and j, we make a list $R_{(i,j)}$ to reduce the lookups in computing $F(\cdot,i,j)$, and each list $R_{(i,j)}$ is updated upon inspecting the next deck. Also, the updates for list $R_{(i,j)}$ can be dovetailed with the updates for lists LD_i and LE_j. Thus, we are able to properly compute entries for tables D, E, and F by looking up entries from our lists, instead of from previously computed table entries. Because PLANAR uses p-part piecewise-linear gap functions, the space used by each list is $O(p)$.

Next, we use a methodology similar to CMSAA's GENERIC_SPLIT that recursively aligns linear chains of T_X individually and recursively splits linear chains of T_X into halves and aligning separately each halve against Y before gluing all results into the final alignment with the assistance of any $R_{(i,j)}$ lists computed over smaller portions. All of this increases runtime by only a constant factor while reducing space usage.

Because the total number of lists of form LD_i, LE_j, and $R_{(i,j)}$ is $O(n^2)$, the total space used by all of the lists is $O(n^2p)$. Furthermore, these lists reduce the overall runtime for the formula mentioned earlier from $O((m+n)n^2m_1 + n^3m_2)$ to $O((\log p)n^2m_1 + n^3m_2)$ because of making only $O(\log p)$ lookups within our lists LD_i, LE_j, and $R_{(i,j)}$ at non-bifurcation nodes. The space is reduced to $O(n^2(p + \log m))$ space, where $O(\log m)$ is used for the recursions over linear chains of T_X. This space usage is asymptotically identical to CMSAA if p is fixed.

Y Secondary Structure Correction

In the event that both X and Y have explicitly established secondary structures, a suitably modified PLANAR works as follows: PLANAR obtains T_X based on X's secondary structure, and aligns T_X to Y in the manner mentioned earlier to get the final alignment tree T_{AX}. PLANAR then obtains T_Y based on Y's secondary structure, and aligns T_Y to X to get the final alignment tree T_{AY}, working exactly the same way as aligning T_X to Y, except that roles of X and Y are interchanged. Using T_{AX}, the alignment obtained using X's secondary structure, and T_{AY}, the alignment obtained using Y's secondary structure, we then proceed to combine them to form the final alignment A in the following way:

We linearize T_{AX} and T_{AY} into respective linear alignments A_X and A_Y. We can make T_{AX} into A_X recursively by performing, for each node u in T_{AX}, a procedure where we place into A_X the left indices of u, then we recursively visit all of u's children nodes (if u is a bifurcation node, we will visit the left node and then the right node), then we place into A_X the right indices of u. This sequence of steps creates the correct linearized alignment A_X. Converting T_{AY} into A_Y uses the same idea. Our goal is to merge A_X and A_Y into a final alignment A that improves upon either of the two computed alignments: A_X or A_Y. Figures 3 and 4 illustrate visually the underlying methodology.

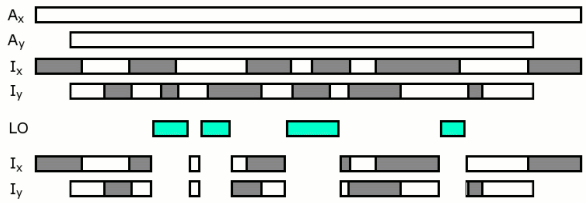

Figure 3: Part 1 of how PLANAR "merges" two alignments A_X and A_Y into the final alignment A. Going from top to bottom, first, I_X and I_Y represent respectively A_X and A_Y fragmented into important and unimportant segments based on alignment scores. Alignment scores are evaluated using the provided gap, and match/mismatch parameters. In the event we are evaluating a bound position, the score is split equally between the two bound positions. The measure of which fragments belong to important segments is performed in a manner similar to that mentioned in SEPA (Gill & Mishra 2006). Below that, LO consists of the identical segments of A_X and A_Y. Next, we trim the segments of I_X and I_Y based on LO.

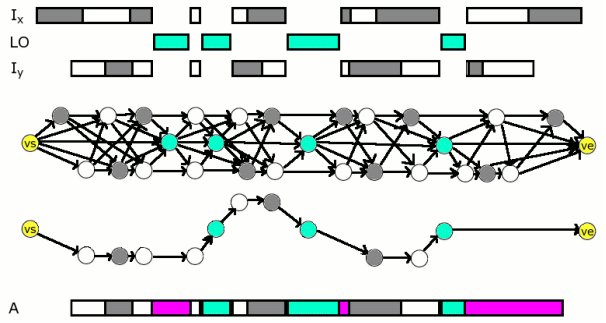

Figure 4: Part 2 of how PLANAR "merges" two alignments A_X and A_Y into the final alignment A. Going from top to bottom, we convert the segments of I_X, LO, and I_Y into nodes, directing an edge from node u to node v only if we can create an alignment where u's segment precedes v's segment. There is one detail not shown here: We assign a weight to each edge (u, v) as the difference between the score for v's segment and the gap penalty for any unused X and Y characters between u and v's alignment segments. We also create a dummy source and a sink node (called respectively vs and ve). For the sake of visual clarity, a few edges have been omitted from this graph. We solve maximum path algorithm over this graph. This is akin to Dijkstra's single-source shortest-path algorithm from(Cormen *et al.* 2001), except that we visit a node only after all of its in-edges have been visited. Below the graph is a drawing using only the nodes and edges involved in the optimal path. Using the segments corresponding to the nodes on this path, plus gaps for any missing X and Y characters, gives us our alignment A.

PLANAR Parameter Optimization

The alignment generated for sequences X and Y vary based on the gap/match-mismatch parameters involved. These parameters can be classified as five variables: $(\alpha, \beta, d, ms, mb)$, where α, β, and d approximate a logarithmic gap function as a p-part piecewise-linear gap function in exactly the same way as in the PLAINS paper(Gill, Zhou, & Mishra 2005), and ms denotes the penalty for an unbound mismatch, and mb denotes the reward for match at a bound position[2]. These five parameters together represent a vector v, and the score of the resulting alignment A from v is denoted by the scalar function $f(v)$. Then, our goal is to find the $v^* = \arg\max f(v)$. This numerical optimization problem is solved in a manner similar to (Gill, Zhou, & Mishra 2005), that is, using either Simulated Annealing or Genetic Algorithms.

PLANAR Empirical Results

Table 1 shows a comparison of alignments for PLANAR and RSMATCH evaluated by SEPA(Gill & Mishra 2006) over biologically related sequences using unadjusted r and t values, and ζ' values, with $\rho = 0.9$. Note that p-values are a measure of reliability for alignments, with ζ' acting as a reliability measure for all important segments of an alignment identified by SEPA, with higher values corresponding to a more reliable alignment. Because of the lower primary structure sequence similarities present in these RNA sequences versus the DNA sequences typically examined by SEPA, we chose to use $\rho = 0.9$ instead of the typical $\rho = 0.5$ value used by SEPA, in order not to risk missing the most important segments[3]. Also, there is a loss of precision involved in the reporting of ζ' values, which in some cases "appear" to be the same for PLANAR and RSMATCH, even if they are not exactly the same. The rnase experiments use Delta/Epsilon Purple Bacteria RNase P sequences from the RNaseP database, and the telomerase experiments use ribonucleoprotein reverse transcriptase synthesizing telomeric DNA found from the RFAM database with accession number RF00025. For further information regarding the sequences used, consult Table 2.

Test Name	PLANAR			RSMATCH		
	t	r	ζ'	t	r	ζ'
rnase.1_2	204.67	3	3.37	118.58	1	5.99
rnase.1_3	134.35	2	4.53	85.55	1	5.99
rnase.1_4	92.53	1	5.98	34.37	2	4.51
rnase.1_5	92.82	2	4.51	57.61	2	4.51
rnase.2_3	184.24	2	4.56	101.35	2	4.53
rnase.2_4	111.35	3	3.27	18.12	2	4.51
rnase.2_5	89.06	2	4.51	41.37	1	5.98
rnase.3_4	78.80	1	5.98	16.17	1	5.98
rnase.3_5	120.76	2	4.51	42.88	1	5.98
rnase.4_5	104.17	1	5.98	93.08	2	4.49
telomerase.1_2	29.58	2	4.47	14.89	2	4.47
telomerase.1_3	79.23	2	4.48	24.67	2	4.48
telomerase.2_3	17.06	1	6.10	2.60	1	6.10

Table 1: Shown here for PLANAR and RSMATCH are the r, t, and ζ' values obtained from aligning noncoding RNA sequences of lengths between 100 and 200 nucleotides (nts) where the pairs are biologically related, with correlated secondary structures, but poor correlations within their primary structures. r denotes the number of important segments identified, and t and ζ' respecitvely denote the total score and the reliability measure for these important segments. Note that ζ' adjusts for sequence lengths, as described in the SEPA paper (Gill & Mishra 2006).

Note from table 1 that, just as with the results discussed in (Gill & Mishra 2006), PLANAR may occasionally fail to yield the results of least coincidental probability for reasons similar to those explaining the behavior of PLAINS. Capturing the biology faithfully when sequences have expected large gaps and low similarities causes PLANAR to aggressively align as many regions as possible, raising its r and t values, with an increase in the r value high enough to ad-

[2]We assume all unbound matches and bound mismatches are each given a reward of 1.

[3]Although SEPA does not account for secondary structure in its p-values or ζ values, the technique used in obtaining alignment A from A_X and A_Y could easily fix this problem. However, this approach would require a separate analysis of the important segments for that case.

versely affect its ζ' value. As a result, if we fix r for PLANAR and RSMATCH, the r segments generated by PLANAR is seen to have smaller individual coincidental probabilities.

Using fixed match/mismatch and gap parameters, for aligning a pair of RNA sequences of length 200 bp, PLANAR takes roughly a minute, about the same amount of time that PLAINS would use for a pair of DNA sequences of length 12 Kb. PLANAR runs about 3 times slower than RSMATCH. The extra time used by PLANAR is usually devoted to the complex task of aligning the piecewise-linear gap functions (compared to the much simpler linear gap functions used in RSMATCH), as well as the time involved in merging two alignments A_X and A_Y.

Conclusions and Open Problems

PLANAR holds significant promises. The evidence in favor of its power are many: It caught stronger correlations than its competitions, achieving an effect similar to PLAINS, except that it works in the more complex domain of RNA sequences. It combines the structure information from both sequences. It incorporates more general gap distributions without paying through a heavy computational complexity for this flexibility. Furthermore, we note that aggressively incorporating too many segment pairs into an alignment can corrupt the overall result with false positives, in spite of an apparent improvement in the total score, as illustrated by PLANAR. However, if we select only the best r segments from an alignment, the strength of PLANAR becomes obvious, since its r segments are less coincidental than its competition, and have higher scores, and hence better ζ' values.

Also, it has become rather apparent that some upward scaling is essential if PLANAR is to be run over sequences with more than 2000 nucleotides. PLANAR is only suitable for sequences of lengths up to 2000 nucleotides due to the higher time and space needed for it to achieve similar power as PLAINS while additionally incorporating secondary structures.

Possible future extensions are in many dimensions. The most important one focuses on aligning large sequences (of megabases of nucleotides), and then using PLANAR to refine alignments over smaller areas, where localized alignments and possibly secondary structures analysis can be performed. With this, PLANAR could become an ideal tool for aligning rRNA's or tRNA's to a genome. Another possible extension involves incorporating a model to learn expected alignments over various species, as opposed to just merely approximating the best gap/mismatch parameters. We can further improve SEPA by accounting for secondary structures in its p-values for RNA alignments.

In addition, PLANAR at the moment assigns to unbounded positions a reward of 1 for match and penalty of m_s (specified by user) for mismatch, and assigns to bound positions rewards of 1 for mismatch and m_b (specified by user) for match. It may be useful to have a more informative scoring matrix to assign different scores to different types of matches/mismatches in these cases. Note that the algorithm and its complexity will remain unchanged, only we need to explore how to model these parameters in a meaningful evolutionary context.

Name	First Sequence	Second Sequence
rnase.1_2	D.desulfuricans	D.vulgaris
rnase.1_3	D.desulfuricans	G.sulfurreducens
rnase.1_4	D.desulfuricans	C.jejuni
rnase.1_5	D.desulfuricans	H.pylori-26695
rnase.2_3	D.vulgaris	G.sulfurreducens
rnase.2_4	D.vulgaris	C.jejuni
rnase.2_5	D.vulgaris	H.pylori-26695
rnase.3_4	G.sulfurreducens	C.jejuni
rnase.3_5	G.sulfurreducens	H.pylori-26695
rnase.4_5	C.jejuni	H.pylori-26695
telomerase.1_2	telo. 1: AF417611/283-441	telo. 2: U10565/50238
telomerase.1_3	telo. 1: AF417611/283-441	azeAF417612/231392
telomerase.2_3	telo. 2: U10565/50238	azeAF417612/231392

Table 2: Sequence details for the RNA alignments computed. All the sequences are retrieved from CARNAC website [http://bioinfo.lifl.fr/carnac/]. The rnase experiments all involve RNase P RNA. Note: telo. is an abbreviation for telomerase.

Sequence Details

Shown in Table 2 are further details for the sequences used to compare PLANAR against RSMATCH.

References

Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. 2001. *Introduction to Algorithms, 2nd Edition*. MIT Press.

Eddy, S. 2002. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an rna secondary structure. *BMC Bioinformatics* 3.

Gill, O., and Mishra, B. 2006. Sepa: Approximate non-subjective empirical p-value estimation for nucleotide sequence alignment. *Lecture Notes in Comp. Sci.* 3992:638–645.

Gill, O.; Zhou, Y.; and Mishra, B. 2005. Aligning sequences with non-affine gap penalty: Plains algorithm, a practical implementation, and its biological applications in comparative genomics. *Series in Math. Bio. and Medicine* 8. An unabridged version can be found at: `http://bioinformatics.nyu.edu/~gill`.

Liu, J.; Wang, J.; Hu, J.; and Tian, B. 2005. A method for aligning rna secondary structures and its application to rna motif detection. *BMC Bioinformatics* 6.

Miller, W., and Myers, E. 1988. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology* 50:97–120.

Rivas, E., and Eddy, S. 1999. A dynamic programming algorithm for rna structure prediction including pseudoknots. *J. Mol. Biol.* 285:2053–2068.