Structural alignment of RNA with FOLDALIGN

Havgaard, Jakob Hull

Structural alignment of RNA with FOLDALIGN

Ph.D. thesis

Jakob Hull Havgaard

Division of Genetics and Bioinformatics
Department of Basic Animal and Veterinary Science
Faculty of Life Sciences
University of Copenhagen
Copenhagen 2007

Contents

Preface

This Ph.D. thesis is the result of my Ph.D. project done at The Royal Veterinary and Agricultural University[1]. The project started May first 2002. It was founded by a grant from the Danish Technical Research Council (STVF, FTP) with additional computer time founded by the Danish Center for Scientific Computing (DCSC).

I would like to thank my supervisor Jan Gorodkin for lots of good feedback (and feed forward) during the project. You always had time to give some extra advise. I would also like to thank the people I collaborated with during the project.

I would also like to thank my wife Merete Havgaard for being patient with me and for doing a lot of proof reading.

[1]By now Faculty of Life Sciences at University of Copenhagen

Publications

Accompanying papers

FOLDALIGN

Havgaard J. H., Lyngsø R. B., Stormo G. D., and Gorodkin J.
Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%.
Bioinformatics 21(9):1815–1824. 2005

Havgaard J. H., Lyngsø R. B., and Gorodkin J.
The FOLDALIGN web server for pairwise structural RNA alignment and mutual motif search.
Nucleic Acids Research 33(Web Server issue):W650–653. 2005
Includes the website `http://foldalign.ku.dk`

Havgaard J. H., Torarinsson E., and Gorodkin J.
Fast pairwise local structural RNA alignments by pruning of the dynamical programming matrix.
Submitted 2007

Torarinsson E., Havgaard J. H., and Gorodkin J.
Multiple structural alignment and clustering of RNA sequences.
Accepted for publication in *Bioinformatics* 2007

Torarinsson E., Sawera M., Havgaard J. H., Fredholm M., and Gorodkin J.
Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure.
Genome Research 16(7):885-889. 2006

MicroRNA

Gorodkin* J., Havgaard* J. H., Enstero M., Sawera M., Jensen P., Öhman M., and Fredholm M.
MicroRNA sequence motifs reveal asymmetry between the stem arms.
Computational Biology and Chemistry 30(4):249-254. 2006
* These authors contributed equally to this work.

Other papers

Andersen E. S., Lind-Thomsen A., Knudsen B., Kristensen S. E., Havgaard J. H., Larsen N., Sestoft P., Kjems J., and Gorodkin J.
Detection and editing of structural groups in RNA families.
Submitted 2006

Gorodkin J., Cirera S., Hedegaard J., Gilchrist M. J., Panitz F., Jørgensen C. B., Scheibye-Knudsen K., Arvin T., Lumholdt S., Sawera M., Green T., Nielsen B. J., Havgaard J. H., Wang J., Li H., Li R., Liu B., Hu S., Dong W., Li W., Yu J., Wang J., Stærfeldt H., Wernersson R., Madsen L. B., Thomsen B., Hornshøj H., Bujie Z., Wang X., Wang X., Bolund L., Brunak S., Yang H., Bendixen B., and Fredholm M.
Porcine transcriptome analysis based on 97 non-normalized cDNA libraries and assembly of 1,021,891 ESTs.
Submitted 2006

Summary

The main purpose of this Ph.D. project is to further develop the FOLDALIGN algorithm. The algorithm simultaneously predicts a common secondary structure and an alignment between RNA sequences.

The structure of an RNA sequence partly defines its function. A prediction of the secondary structure can aid the prediction of a sequence's function. The secondary structure is a list of the base pairs in the structure. Most often it is further limited to the structure with the most base pairs where any pseudo knots has been removed.

When sequences can be aligned, it gives a hint that they might be related, and they may therefore also have the same or similar function.

It is often difficult to align RNA sequences without using structural information because the primary sequence can relatively easy mutate without changing the structure. This is due to compensatory mutations where one or both nucleotides in a base pair is mutated without breaking the base pair.

Earlier versions of FOLDALIGN align two or more sequences [Gorodkin et al., 1997b,c, 2001a,b]. A high positive score is given when a base pair is conserved, and a negative score is given when a single stranded nucleotide is substituted for another, or a gap is inserted. To save time the structure is limited to non-bifurcated structures. A bifurcated structure is a structure made up of two smaller independent structures.

The pairwise FOLDALIGN algorithm has been improved in several ways. In Havgaard et al. [2005b] an energy model somewhat similar to the one used in single sequence folding is added. More complicated structures can be aligned as bifurcated structures are allowed. Allowing bifurcating structures slows down the algorithm. To speed up the algorithm a constraint is placed on the bifurcation calculation. Most bifurcated structures can usually be made in more than one way. The constraint limits this to only one way. To reduce memory requirements, while aligning long sequences, one sequence is split into several smaller chunks. Each chunk is aligned to a window on the other sequence. When all nucleotides in the window has been aligned to the nucleotides in the chunk, the window is moved one nucleotide. The new nucleotide is aligned to those in the chunk, and the process is repeated. When all nucleotides in the second sequence have been aligned to all the nucleotides in a chunk, the process starts over with the next chunk. The alignment can be shorter than the window, but not longer. Havgaard et al. [2005a] describes a webserver and presents results for global alignment in the supplementary material.

Havgaard et al. [2007] describes a method for making algorithms like FOLDALIGN run very fast. When two sequences are aligned, a lot of low scoring sub-alignments must be calculated. The method assumes that these bad sub-alignments cannot be part of any good alignment, and

they are therefore pruned away. This is simply done by requiring that an alignment of a given length must have a score above a minimum score, or it is pruned away. Since the alignments which are pruned away cannot be part of the longer alignments, the calculations of the longer alignments do not have to use time taking the bad alignments into account. It is not necessary to keep all the bad alignments in memory which saves a lot of memory. To reduce memory consumption even more the constraint on the bifurcation calculation is used to limit the amount of information which must be stored. When this reduction is used, there is not enough information for the algorithm to perform the backtracking needed to retrieve the structure and the alignment. More information must therefore be stored. In an attempt to limit the amount of extra information which must be stored, a pre-backtrack is made which locates all the bifurcation points. These are then used to split the structure into non-branched subsegments. These subsegments are usually much shorter than the full structure and can therefore be backtracked without using extra memory. Unfortunately there is no guaranty that this method will use less memory.

Torarinsson et al. [2007] describes a method for making global multiple alignments. It is based on the PMcomp algorithm [Hofacker et al., 2004]. It can cluster the sequences based on their alignment scores and then find a consensus structure for each cluster. The consensus structure is found by aligning base pairing probability matrices. These are calculated using either the McCaskill [McCaskill, 1990] or FOLDALIGN algorithms. The constraints used in the pairwise algorithm are also used in the multiple version.

In Torarinsson et al. [2006] FOLDALIGN is used to align RNA sequences from human and mouse. The aligned regions are just upstream or downstream of sequences which can be aligned using normal sequence alignment methods. The regions are terminated by either gaps, repeats, or new alignments. The regions themselves cannot be aligned using conventional methods. For details see Figure 1 in Torarinsson et al. [2006] on page 99. Sequence pairs were made from ten different chromosomes. 1297 good alignments were found. If all chromosomes had been used, an estimated 3600 good alignments would have been found. This is approximately 1800 more than what would have been expected by chance. 32 out of 36 top candidates tested gave a good positive result when tested with PCR. 12 of those which gave good results by PCR, were tested using Northern blots, and four of these gave good results.

The final paper, Gorodkin et al. [2006], is about the family of RNA genes called microRNAs. In the paper it is shown that there is a difference in sequence motif depended on whether the mature microRNA is located on the left or right side of the predicted precursor hairpin loop.

Sammendrag på dansk

Hovedformålet med dette ph.d.-projekt er at videreudvikle FOLDALIGN-algoritmen. Algoritmen er beregnet til samtidigt at forudsige en fælles, sekundær struktur for RNA-sekvenser og rette sekvenserne ind efter hinanden[2].

Strukturen af en RNA-sekvens er med til at bestemme dens funktion. En forudsagt, sekundær struktur kan derfor være med til at forudsige funktionen af en sekvens. Den sekundære struktur af en RNA-sekvens består i en liste af de basepar, som et molekyle med denne sekvens danner. Som oftest begrænses den endvidere til den struktur, som har flest basepar, når pseudo-knuder fjernes.

Hvis to sekvenser kan alignes godt, så giver det et vink om, at de muligvis er relaterede. Hvis en menneskesekvens med kendt funktion kan alignes til en grisesekvens, så er der grund til at tro, at de to sekvenser har de samme eller relaterede funktioner.

RNA-sekvenser kan oftest ikke alignes ordentligt uden brug af struktur, fordi den primære sekvens af et RNA-molekyle relativt nemt kan muteres, uden at strukturen af molekylet ændres væsentligt. Dette skyldes kompenserede mutationer, hvor en eller begge nukleotider i et basepar skiftes ud med andre nukleotider, sådan at baseparret bevares.

De tidligere versioner af FOLDALIGN aligner to eller flere sekvenser [Gorodkin et al., 1997b,c, 2001a,b]. Der gives en høj positiv score, når et basepar bevares, og en negativ score når enkeltstrengede nucleotider ikke er bevaret, eller der indsættes mellemrum i sekvensen. For at spare tid kan strukturen ikke indeholde bifurkationer. En bifurkeret struktur er en struktur, der består af to mindre, uafhængige strukturer.

De nye versioner af algoritmen, som er udviklet under dette ph.d.-projekt, er forbedret på flere måder. I Havgaard et al. [2005b] tilføjes der en energimodel, der minder om den, der bruges til enkelt sekvens-foldning. Mere komplicerede strukturer kan findes, fordi bifurkerede strukturer nu er tilladt. Brugen af bifurkerede strukturer gør algoritmen langsommere. For at spare tid er der indført en begrænsning, hvor den bifurkerede struktur kun beregnes på en af flere måder. For at begrænse hukommelsesforbruget ved lange sekvenser bliver den ene sekvens klippet op i mindre stykker, som alignes til et vindue fra den anden sekvens. Når alle nukleotider i vinduet er alignet til stykket fra den første sekvens, så rykkes vinduet et nukleotid. Det nye nukleotid alignes til stykket, hvorefter vinduet rykkes igen. Når hele den anden sekvens er blevet alignet til stykket fra den første sekvens, så starter processen forfra med det næste stykke fra den første sekvens. En alignment kan godt være kortere, men ikke længere

[2]At rette sekvenser ind efter hinanden kaldes ofte at *aligne* dem. Resultatet af at aligne to eller flere sekvenser kaldes en *alignment*.

end vinduet. Havgaard et al. [2005a] beskriver en ny webserver for programmet. Desuden beskrives resultater for global alignment i Supplementary material.

Havgaard et al. [2007] beskriver en måde til at få programmer som FOLDALIGN til at køre hurtigt. Metoden udnytter, at mange af de delalignmenter, som beregnes undervejs, er meget dårlige. Ved at kræve at alle alignmenter med en vis længde skal have en score over et vist minimum, begrænses antallet af delalignmenter. De alignmenter, der ikke lever op til minimumskravet, kan ikke være en del af længere alignmenter, og der skal derfor ikke bruges tid på at tage de dårlige delalignmenter med i beregningen af de længere alignmenter. Det er ikke nødvendigt at gemme de dårlige delalignmenter, og derfor spares der også meget hukommelse. For yderligere at spare hukommelse udnyttes begrænsningen på det antal måder, en bifurkteret struktur udregnes på til kun at gemme den mest nødvendige information. Denne ekstra besparelse betyder, at den del af algoritmen, som går tilbage gennem den lagrede information for at finde alignmenten og dens struktur, ikke har nok information. Der skal altså lagres mere information for at kunne lave denne tilbage-sporing. I et forsøg på at begrænse mængden af information, der gemmes, bruges der en "del og hersk-strategi". Først laves en tilbage-sporing, der fastlægger alle bifurkations-punkter. Disse punkter bruges til at splitte strukturen op i mindre dele, som ikke indeholder bifurkationer. Fordi de mindre dele som oftest er væsentligt mindre end hele strukturen, skal der oftest bruges mindre hukommelse, når de tilbage-spores. Der er dog ingen garanti for, at strategien bruger mindre hukommelse end en simpel tilbage-sporing.

De første versioner af FOLDALIGN kunne lave alignmenter med flere sekvenser, mens version 2 kun kan lave dem parvist. FOLDALIGNM [Torarinsson et al., 2007] kan lave globale alignmenter med flere sekvenser. Sekvenserne bliver samlet i grupper efter deres score, og for hver gruppe findes der en fælles struktur. Strukturen findes ved at aligne basepar-sandsynligheds-matricer beregnet med McCaskill [McCaskill, 1990] eller FOLDALIGN-algoritmerne.

I Torarinsson et al. [2006] bruges FOLDALIGN til at sammenligne RNA-sekvenser fra menneske og mus. Sekvensstykkerne er udvalgt sådan, at i den ene ende kan nabosekvenserne alignes, mens nabosekvensen i den anden enden er et mellemrum, et gentagelseselement eller en ny alignment. Sekvenserne i selve sekvensstykket kan ikke alignes med standardmetoder. Se figur 1 i Torarinsson et al. [2006] på side 99. Der blev brugt sekvenser fra ti forskellige kromosomer. Der blev fundet 1297 signifikante alignmenter. Det svarer til $\sim$3600 kandidater, hvis alle kromosomer blev brugt. Ca. 1800 af disse kan ikke forklares ud fra, hvad der ville forventes ved et tilfælde. Af 36 topkandidater, der blev testet i laboratoriet med PCR, gav de 32 et positivt resultat. 12 af de kandidater, der gav positive PCR-resultater, blev også testet med Northern blots, og fire af disse blev genfundet.

Den sidste artikel, Gorodkin et al. [2006], handler om den familie af RNA-gener, der hedder microRNAer. I denne vises det, at sekvensmotiver for microRNA er forskellige, alt efter om microRNA-sekvensen ligger på højre eller venstre side af den struktur, som microRNAen kommer fra.

Chapter 1

Introduction

The main purpose of this Ph.D. project is to develop a better method for local structural alignment of RNA sequences. The project is based on the FOLDALIGN algorithm [Gorodkin et al., 1997b,c, 2001b,a] which is the first simplified implementation of the general Sankoff algorithm [Sankoff, 1985] for structural alignment of RNA.

The first section in this chapter gives a short introduction to RNA sequences. The second section introduces RNA alignment and the results of the FOLDALIGN papers included in this thesis [Havgaard et al., 2005a,b, 2007, Torarinsson et al., 2007, 2006]. The final section introduces micro-RNAs (miRNA) and the results of the miRNA paper [Gorodkin et al., 2006].

The second chapter is focused on the pairwise FOLDALIGN algorithm. It starts with a description of the two new implementations (2.0 and 2.1) and their time & memory requirements. The second section describes the energy model. The next section describes the heuristics used in the algorithm. The fifth section describes the parameters used. It is followed by a section about selecting and evaluating the significance of the local alignments. The final section describes the datasets used to train and test the algorithm.

The third chapter gives a short description of the results from each paper, and some concluding remarks are made.

The appendix describes the recursion used by FOLDALIGN.

The six papers make up the remaining part of the thesis.

1.1 RNA sequences

An RNA sequence is a polymer of nucleotides linked to each other at the $3'$ to $5'$ positions. This leaves the $5'$ end free on the first nucleotide and the $3'$ end free on the last nucleotide. The sequence is therefore said to be in the $5'$ to $3'$ direction. Each nucleotide consists of a ribose and phosphate backbone part and a base. Usually the base is an Adenosine, a Cytosine, a Guanine, or an Uracil. The polymer is flexible and can bend back on itself allowing the bases of the nucleotides to interact through hydrogen bonding. The bonding is further stabilized through stacking. The canonical base pairs are *A* - *U* and *G* - *C*. The wobble base pair *G* - *U* is usually also included among the RNA base pairs.

The primary structure of an RNA sequence is the list of the nucleotides in the $5'$ to $3'$ direction. The secondary structure is a list of the base pairs found in the three-dimensional structure of the molecule. Often the secondary structure is limited to the list of base pairs when pseudo knots have been removed. If the nucleotides at positions i and j in a molecule base pair, and the nucleotides at position m and n also base pair and $i < m < j < n$, then one of the base pairs is part of a pseudo knot. Secondary structure prediction based on a single sequence started with [Tinoco Jr. et al., 1971, Tinoco Jr. et al., 1973]. Efficient dynamic programming algorithms were introduced by [Waterman, 1978, Waterman and Smith, 1978, Nussinov et al., 1978, Nussinov and Jacobson, 1980]. Today the most widely used programs are Mfold [Zuker, 2003] and RNAfold from the Vienna package [Hofacker et al., 1994].

Traditionally there were four different types of RNAs: Messenger RNA (mRNA) which is the RNA intermediate between a protein and its DNA sequence. The ribosome RNAs (rRNA) which are the primary molecules in the translation of the mRNAs into proteins. The transfer RNAs (tRNA) catch amino acids and bring them to the right positions in protein sequences under construction. Finally there was also “a few other types of RNAs”. The important RNAs were the mRNAs because they are directly related to the proteins, and proteins did everything except those things done by rRNAs, tRNAs, and “a few other types of RNAs”. This view is being revised. Currently there are still four main types of RNAs: The mRNAs, the RNA elements, the non-coding RNAs (ncRNA), and “a few other RNAs”. The mRNAs are still the RNA intermediate between a protein and its DNA sequence. An RNA element is an RNA structure in an RNA molecule which has a function which is not part of the primary function of the molecule. This can for example be UTR elements affecting the translation rate of a mRNA [Winkler, 2005], a self splicing intron [Woodson, 2005], or a seleno cysteine insertion element [Walczak et al., 1996]. The ncRNAs are RNA genes like the “old” rRNA and tRNA. These RNA molecules have functions in a fashion similar to proteins. Often the function involves base pair interactions between different molecules. The ncRNAs are found both as independent genes or inside the introns of other genes. In the digital RNA theory by Mattick [2004] some RNA sequences are believed to be adapter molecules between regulatory complexes and the sequences being regulated. The “a few other RNAs” category is still around since the RNA field is expanding rapidly, and new types of molecules may be discovered.

The view of the roles of RNA is changed by a combination of high throughput experimental techniques, bioinformatics, and renewed interest in the RNA field. The high throughput experimental techniques including sequencing and arrays have to a large extend been pioneered by the RNomics group [Huttenhofer et al., 2002] and the RIKEN group [Suzuki and Hayashizaki, 2004]. Huttenhofer and Vogel [2006] reviews the experimental techniques. Some of the bioinformatic methods will be discussed in the next section.

1.2 RNA alignment

Alignment of sequences has been one of the most useful disciplines in bioinformatics. An alignment helps transforming knowledge about one sequence into knowledge about other sequences. Even when there is no knowledge about any of the sequences in an alignment, the information that sequences can be aligned, can be used to select sequences for further investigation. Efficient methods for alignment of sequences by sequence similarity have been around since dynamic programming was introduced into the field of bioinformatics. For a historical account see [Sankoff, 2000].

Alignment of non-protein-coding-RNA sequences is complicated by the fact that certain parts of the primary structure can be mutated without changing the structure of the molecule significantly. It is often only important that the nucleotides at two positions in the sequence base pair, the types of nucleotides at the positions matter less. A $G - C$ base pair can for example be replaced by an $A - U$ base pair. Wobble base pairs $G - U$ make it even easier for a sequence to change since they allow the nucleotides of a base pair to mutate in separate events without changes to the structure. Two homologous RNA sequences can therefore have dissimilar primary sequences while retaining similar structures. It is therefore desirable to include structural information when aligning RNA sequences. To simplify algorithms only secondary structure information (without pseudo knots) is normally used.

RNA alignment and structure prediction were combined in the Sankoff algorithm [Sankoff, 1985] which describes an algorithm for multiple alignment and common structure prediction. The Sankoff algorithm's time and memory complexity becomes intractable for more than a few sequences. A simplified version of the Sankoff algorithm was implemented in the first version of FOLDALIGN [Gorodkin et al., 1997b,c, 2001a,b]. From pairwise alignments it uses greedy algorithms to build the multiple alignment. The pairwise alignment maximizes a score where conserved base pairs are given a positive score even when the nucleotides have changed. An extra score can be added when several base pairs are nested in a stem. Single stranded nucleotides are given a positive score when they are conserved between the two sequences, and a negative score when the nucleotides are not conserved.

In general there seems to be two main classes of algorithms which are aimed at making alignments of RNAs or predicting common folds. The energy based algorithms like FOLDALIGN, Dynalign [Mathews and Turner, 2002], RNAz/RNAalifold [Washietl et al., 2005b, Hofacker et al., 2002], Pmcomp [Hofacker et al., 2004], RNAcast [Reeder and Giegerich, 2005], and Cofolga [Taneda, 2005] use energy minimization as the basis for the algorithm. The stochastic methods like Cove [Eddy and Durbin, 1994], Infernal [Eddy, 2002], Stemloc [Holmes, 2005], Consan [Dowell and Eddy, 2006], QRNA [Rivas and Eddy, 2001], Evofold [Pedersen et al., 2006], and PFOLD [Knudsen and Hein, 2003] are based on the ideas of stochastic context free grammars, SCFGs, see also [Durbin et al., 1998]. CMfinder [Yao et al., 2006] uses expectation maximization in a Bayesian frame work. There are several other methods as the field is growing fast. The Wiki (http://wikiomics.org/wiki/List_of_articles#RNA) started by Paul Gardner keeps track of the different methods. Most methods are currently aimed at making global multiple alignments or structure predictions. Most of the methods are very slow and use huge amounts of computer memory. To make the methods usable heuristics and/or simplifications are used. Popular heuristics include pre-folding of the sequence which limits the number of base pairs in the sequence, or similarity anchoring which requires that certain positions in the sequences are aligned. Biological simplifications like ignoring multibranched loops or collapsing structures into a stem representation are also used.

The current version of the pairwise FOLDALIGN algorithm is presented in this thesis. It has several advantages compared to most of its competitors. It can make both local and global alignments. It is fast and has a low memory requirement due to the use of heuristics. The heuristics used do not remove the comparative information and do not require sequence similarity. Even though sequence similarity is not required it can be used if present. A weakness of the current pairwise algorithm is that it cannot align multiple sequences. A first step in making a new local multiple alignment method is the method for making global multiple alignments presented in [Torarinsson et al., 2007]. In addition to the FOLDALIGN algorithm a script which evaluates the significance of the local alignments, is also described.

The aim of the Havgaard et al. [2005a,b, 2007] papers are to improve the predictive performance and the time and memory requirements of FOLDALIGN. Havgaard et al. [2005b] introduces a light weight energy model and removes the stem-only simplification from the algorithm. The stem-only simplification was replaced by a heuristic which limits the number of ways a given multibranched loop is calculated to one. A scanning scheme is also implemented which lowers the memory requirement at the cost of doubling the run time. While local alignment and structure predicting are the main foci of the first paper, the second, Havgaard et al. [2005a], introduces a webserver and results for global alignment. Havgaard et al. [2007] introduces the pruning heuristic which significantly speeds up the algorithm and lowers the memory consumption. The heuristic which limits the number of ways a multibranched loop is calculated, is used to lower the memory requirement further. When this heuristic is used, not all of the cells in the dynamic programming matrix passed by the backtrack algorithm contains a value. It is therefore not possible to do a normal backtrack through the matrix. A “divide and conquer” strategy is used to do the realignment and backtracking of the structure in an attempt to try to avoid using too much extra memory. There are also small improvements to the energy model.

The first version of FOLDALIGN can make multiple alignments whereas the second version can only make pairwise alignments. FOLDALIGNM [Torarinsson et al., 2007] is a global multiple alignment algorithm based on the PMcomp algorithm [Hofacker et al., 2004]. It aligns base pair probability matrices based on the McCaskill [McCaskill, 1990] or FOLDALIGN algorithms. Clustering is used to group sequences with similar structures which are then given a consensus structure.

Recently three large scale searches for ncRNAs in vertebrates have been published [Washietl et al., 2005a, Pedersen et al., 2006, Torarinsson et al., 2006]. Washietl et al. [2005a] uses the RNAz algorithm [Washietl et al., 2005b]. RNAz compares the minimum free folding energy of a common fold (with an additional substitution score) of an alignment to the free folding energies of the individual sequences [Washietl and Hofacker, 2004, Hofacker et al., 2002]. A support vector machine (SVM) is used to calculate a Z-score for the alignment. Pedersen et al. [2006] uses the Evofold algorithm to scan multiple alignments of vertebrates for new ncRNAs in humans. Evofold uses a SCFG and a phylogenetic model to evaluate how well the substitutions in a multiple alignment agree with it being a conserved ncRNA structure. Torarinsson et al. [2006] uses FOLDALIGN and is part of this thesis, see page 97. Compared to RNAz and Evofold FOLDALIGN has the advantage that it is local, and it can be used on unaligned sequences. RNAz’s and Evofold’s advantage is that they can work with multiple sequences. The three studies show that the general bioinformatical tools for finding novel ncRNAs have reached a point where they are becoming useful not only for benchmark comparisons, but also for generating new knowledge about biology.

1.3 MicroRNA

The final paper in this thesis is about microRNAs (miRNA). MiRNAs are $\sim$22 nt. long RNA sequences which inhibit the translation of their RNA targets. Targets are selected by base pairing between the miRNA and the target RNAs. Even though the first miRNA gene had already been discovered by others [Lee et al., 1993], the most important ncRNA papers recently are probably the three miRNA papers published in 2001 by Lagos Quintana et al. [2001], Lau et al. [2001], and Lee and Ambros [2001], since they brought a huge amount of spotlight on the ncRNA field.

A miRNA is cleaved out of a pre-miRNA which is a stem loop. In plants the stem can be very long, but in other organisms the stem is $\sim$80 nucleotides long. The miRNA also known as the mature miRNA can be located at either side of the loop. In Gorodkin et al. [2006] it is shown that the sequence motif of mature miRNAs which comes from the left side of the stem, is different from the sequence motif of those which comes from the right side of the stem. This indicates that there might be a difference in the cellular machinery used to process the left and right mature miRNAs. Furthermore the sequence motifs of the different groups of organisms varies indicating that there are also differences in the processing machinery between the different species.

Chapter 2

FOLDALIGN

2.1 Implementations

Two new implementations of FOLDALIGN have been made as part of this project. The second version (2.1) replaces the first version (2.0). The main difference between the two versions is the pruning constraint, see section 2.3.1, and better implementation which makes the program use less memory and run faster. In the following I and K are the sequences being aligned. L_I is the length sequence I. λ is the maximum motif length. δ is the maximum length difference between two subsequences being aligned. i & j are start and end coordinates of a sub alignment from sequence I, and k & l are similar coordinates from the K sequence.

2.1.1 Version 2.0

Version 2.0 of FOLDALIGN is described in the paper Havgaard et al. [2005b]. The main aims for version 2.0 were:

1. Allowing for bifurcated structures
2. Lowering the memory complexity by implementing a scanning scheme. This allows the algorithm to locally align long sequences
3. Improving the pairwise alignment performance by using a better scoring scheme

Many classes of RNA structures contain bifurcations. Constraining the algorithm to non-bifurcated structures as done in the earlier versions [Gorodkin et al., 1997b,c, 2001a,b] significantly limits its performance [Gardner and Giegerich, 2004]. Allowing bifurcating structures changes the time complexity of the algorithm from $O(L_I^2 L_K^2)$ to $O(L_I^3 L_K^3)$ slowing down the algorithm significantly. To alleviate this a new constraint was added which limits the number of times identical structures are calculated to one, see section 2.3.

The scanning scheme splits one sequence into overlapping subsequences and aligns each of these subsequences to the other sequence. Each subsequence overlaps the previous subsequence with λ nucleotides and is typically 2λ nucleotides long (where λ is the maximum motif length possible, see section 2.3). In the other sequence only a λ long section of the dynamic programming matrix is kept in memory. For details see Havgaard et al. [2005b] page 59 and Figure 2 on the same page. The scheme lowers the memory complexity from $O(L_I L_K \lambda\delta)$ to $O(\lambda^3\delta)$ at the maximum cost of doubling the run time.

Initial tests showed that if the algorithm was to have a good performance, a better scoring function than the simple substitution scheme used in the earlier versions Gorodkin et al. [1997c] was needed. Therefore an energy model was added to the algorithm. The energy model uses five different contexts: Hairpin loops, stems, bulge loops, internal loops, and bifurcation loops. The scoring scheme is described in section 2.2.

2.1.2 Version 2.1

Version 2.1 of FOLDALIGN is described in Havgaard et al. [2007]. The main improvements are:

1. Pruning
2. Better memory implementation during:
   (a) Scanning - keeping only the necessary information
   (b) Non branched alignment - Exploiting the lower requirements during scanning
   (c) Backtrack - Using a “divide and conquer” algorithm to try to keep memory consumption low during backtrack
3. Better use of the δ constraint during global alignment
4. Improving the energy model:
   (a) By always scoring single stranded nucleotides external to any base pair in the same way
   (b) By allowing more base pair inserts

The two major problems of the 2.0 version of FOLDALIGN are the time and memory needed to run the program. In the pruning heuristic it is assumed that any subalignment with a score below a given cut off will never be part of any biologically relevant alignment. The algorithm saves time by ignoring such subalignments. Pruning also saves memory space since it is not necessary to store information about subalignments which have been pruned away.

An alignment starting at position i can, with the exception of bifurcation, only be expanded into alignments starting at position i or $i-1$, see the recursion in appendix A. In the case of bifurcation an alignment starting at position i can become part of an alignment starting further downstream. The memory is therefore split into two parts. The short term memory keeps all the needed information about all alignments starting at position i and $i-1$. The long term memory holds the information for alignments with starting positions in the range from i up to $i+\lambda$. Only information about alignments (stems) which can form the right part of a bifurcation, is stored in the long term memory, see Figure 2.11. The short term memory has a memory complexity of

$O(\lambda^2\delta)$, and the long term memory's complexity is $O(\lambda^3\delta)$. While the complexity of the long term memory is the same as the memory complexity in the 2.0 version, the long term memory matrix is much more sparse.

In the non branched alignment case the information in the long term memory is not needed which further lowers the memory requirement during the scanning phase. The memory requirement is that of the short term memory, namely $O(\lambda^2\delta)$.

During backtrack information about all subalignments which will be passed during the backtrack, is necessary. This means that much more information must be stored in the long term memory. To try to avoid raising the memory requirement during backtrack too much a "divide and conquer" scheme is used. First step is a pre-backtrack which realigns the region of interest, but in addition to the stem alignments also bifurcation alignments are kept in the long term memory. An extra pointer to the last bifurcation point passed by an alignment is also kept for all alignments. Finally, also a list of all bifurcation points encountered is kept. By using this information it is possible to find all the bifurcation points in the alignment which is then divided into non bifurcated stem segments. Each stem segment is realigned and backtracked keeping the full long term memory. This is somewhat similar to the "divide and conquer" strategy used by Eddy [2002]. The memory is saved since each stem segment usually is much shorter than the full alignment. The run time complexity of the pre-backtrack realignment is identical to the realignment needed without this scheme. The run time of the stem segment realignments is relatively short as these are non branched $O(L_{segment}^2\delta^2)$. In most cases this scheme allows the backtrack to be performed without using more memory than during the initial scan. But a clear example where it does not work, is in the case of non branched alignments. A cubic space model, similar to the linear space models used in sequence alignment, could be used to ensure that the memory consumption during backtrack of a stem segment stays below a given cut off [Myers and Miller, 1988, Hirschberg, 1975].

During backtrack and/or global alignment the δ constraint (see section 2.3) can be used to reduce the run time and memory consumption even further. The δ constraint combined with the "end to end" requirement of global alignment constrains the k position to a 4δ wide band. This is similar to the M constraint used in Dynalign [Mathews and Turner, 2002].

The energy model used in version 2.0 has no consistent way of handling single stranded nucleotides outside any base pairs. They are scored as hairpin loop, internal loop, or bifurcation loop nucleotides. In the 2.1 version they are always scored as bifurcation loop nucleotides. An simplified overview of the energy model can be seen in Figure 2.5.

In Dynalign, [Mathews and Turner, 2002], a base pair can only be inserted in a stem when it is surrounded by two conserved base pairs. Version 2.0 of FOLDALIGN uses a similar insert model. In 2.1 this requirement is relaxed so that only the first base pair must be conserved, and that the stem contains at least two conserved base pairs. In a future version the requirement for the first base pair to be conserved will probably be removed.

Dangling ends which are single stranded nucleotides in multibranched loops or external loops which are located next to a base paired nucleotide, no longer get a stacking bonus in version 2.1. The predictive performance gained by using dangling ends is small compared to the complexity which they add to the algorithm.

	Local	BT/Global	No_branch local	No_branch BT/Global
2.0 Time complexity	$O(L_I L_K \lambda^2 \delta^2)$	$O(L_I^3 L_K \delta^2)$	$O(L_I L_K \lambda \delta)$	$O(L_I^2 L_K \delta)$
2.1 Time complexity	$O(L_I L_K \lambda^2 \delta^2)$	$O(L_I^3 \delta^3)$	$O(L_I L_K \lambda \delta)$	$O(L_I^2 \delta^2)$
2.0 Memory complexity	$O(\lambda^3 \delta)$	$O(L_I^2 L_K \delta)$	$O(\lambda^3 \delta)$	$O(L_I^2 L_K \delta)$
2.1 Memory complexity	$O(\lambda^3 \delta)$	$O(L_I^2 \delta^2)$	$O(\lambda^2 \delta)$	$O(L_I^2 \delta^2)$

Table 2.1: Time and memory complexity. “Local” is the local alignment scan case. “BT/Global” is the backtrack and global alignment cases. “No_branch” are the non-bifurcated cases.

2.1.3 Time and memory

The time and memory complexities can be seen in Table 2.1. Figures (2.1) and (2.2) show the average time needed to align one of the eight SRP sequence pairs, see section 2.6. The sequences are 1000 nucleotides long, and each contains one ∼300 nucleotide long SRP motif. The runs were made using $\delta = 25$ and $chunk\_size = 1000$. The machines used have Intel XEON 2.4 GHz processors, and the source code had been compiled using the gcc 3.4.4 compiler. The “shuffled data” curve was made with shuffled versions of the same sequences. The two “2.1, Pruning” curves are the same in the two figures. From the figures it is clear that the pruning constraint speeds up the calculations significantly. The “2.1, No pruning” curve was not extended beyond $\lambda = 300$ due to time constrain, whereas the pruning curves could be easily extended. It can also be seen that the 2.1 implementation is faster than the 2.0 implementation. Figure (2.2) shows that the pruning constraint is slightly less effective when the sequences being aligned share a common motif, which is to be expected.

Figure (2.3) shows the average memory used for the same alignments as in Figures (2.1) and (2.2). In this figure it can be seen that the memory requirement is a major limitation for the 2.0 version of the algorithm. It is also clear that the pruning constraint also has a very good effect on the memory usage. This figure also shows that the pruning constraint is slightly more efficient when there is no common motif.

Figure (2.4) shows the average time required to align two shuffled sequences of the same length using a λ equal to the sequence length ($\delta = 25$). The machines used have Intel Xeon 5150 Woodcrest 2.66 GHz CPUs, and the source code had been compiled using the gcc 4.1.0 compiler. Note that the machines used to make this figure are different from those used to make the previous figures, and that the figures therefore are not directly comparable. Without pruning the time complexity is expected to be $O(\lambda^4\delta^2)$. For these eight sequences it appears to be $O(\lambda^{\sim 2.4}\delta^2)$. The run time of the algorithm scales significantly better when the pruning constraint is used.

For global alignment the time and memory results are less clear (data not shown). The memory usage is greatly influenced by the structure being aligned. When the structure is unbranched, and the sequences are short, the 2.0 and the 2.1 no pruning versions of the algorithm require approximately the same amount of memory. The pruning constraint lowers the requirement, but not as much as in the local alignment case, see section 2.3.1. The run time advantage gained by using pruning as a function of the δ parameter can be seen in Figure (2.12).

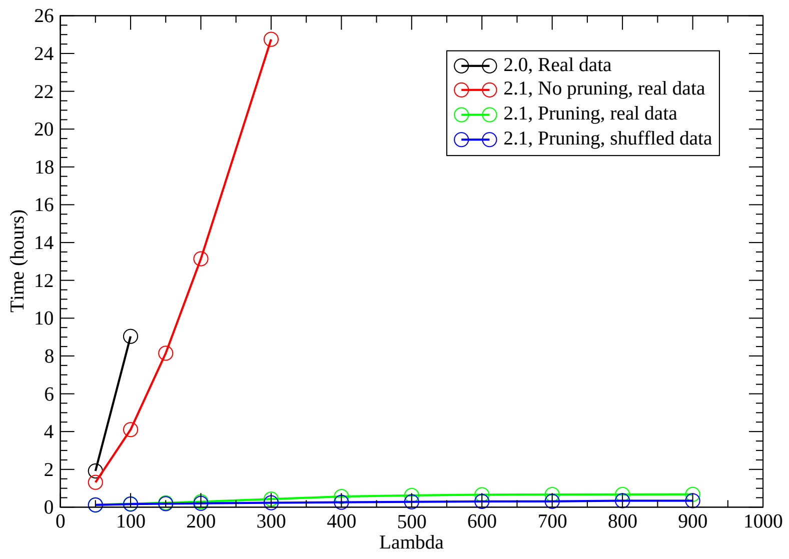


Figure 2.1: Average run time as a function of motif length - λ. The 1000 nt. long SRP data set was used. $\delta = 25$ and $chunk\_size = 1000$. The new implementation is faster than the old even without pruning. The time needed to align unrelated sequences was measured by aligning shuffled SRP sequences. Figure (2.2) shows the pruning curves in more detail.

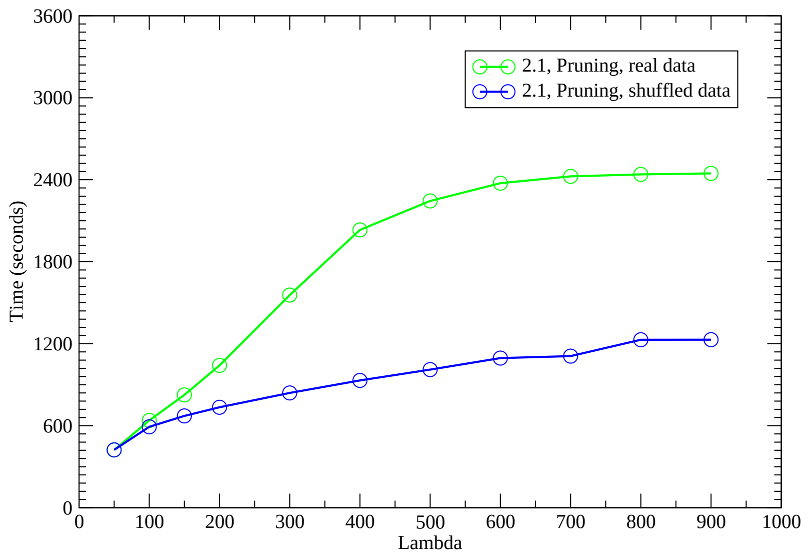


Figure 2.2: Average run time as a function of motif length - λ. The run time increases when a motif is present. The "real data" contains SRP sequences with a length of ~ 300. The "shuffled data" contains the same sequences shuffled.

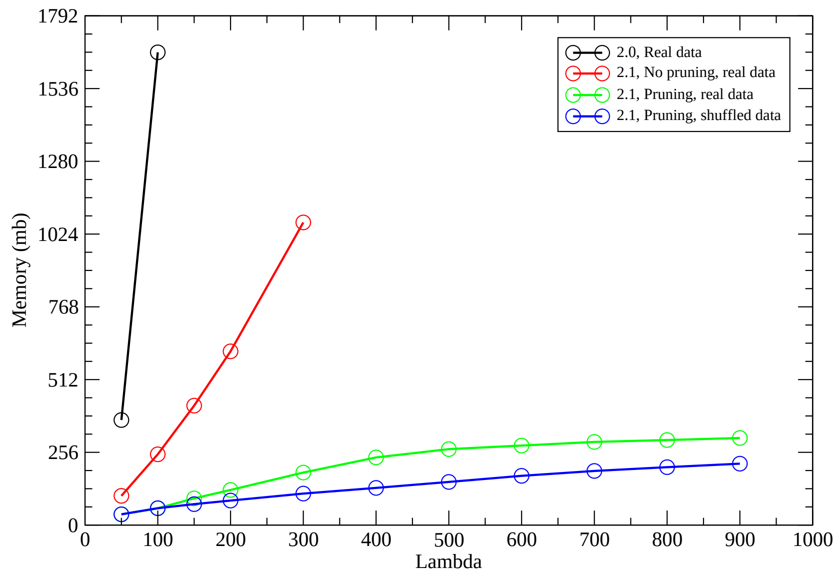


Figure 2.3: The average memory requirement as a function of motif length - λ. The 2.1 implementation clearly uses a lot less memory than the 2.0 implementation even without pruning.

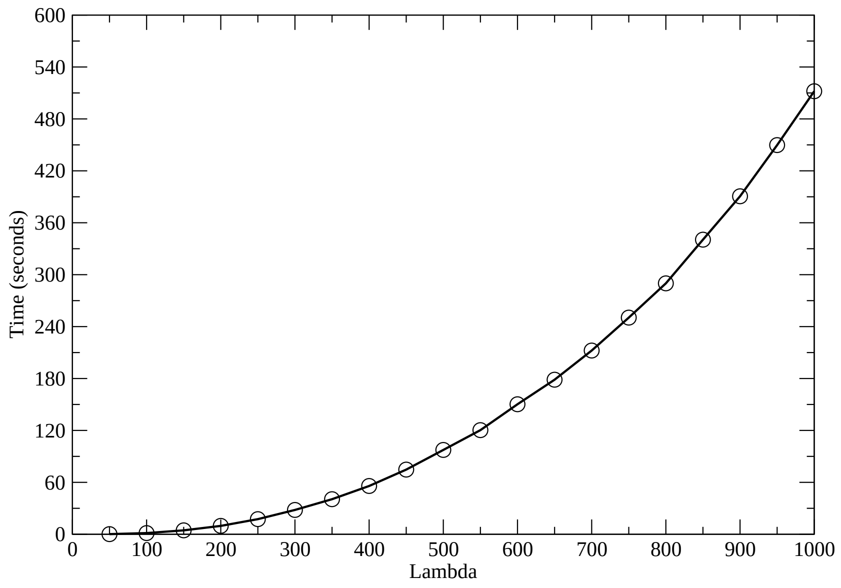


Figure 2.4: Average run time as a function of the lengths of the input sequences. In these runs shuffled sequences with lengths equal to λ were aligned. Note that the machines used to make this figure and Figures (2.1) and (2.2) are different from each other.

2.2 Scoring scheme

The scoring scheme has five contexts: Hairpin-loops, stems, bulge-loops, internal-loops, and bifurcated-loops. An alignment always starts in the hairpin loop context. In FOLDALIGN version 2.0 the alignment can end in any context except the bulge context. In version 2.1 the alignment can only end in either the stem or bifurcation context. Figure 2.5 shows an overview of the energy model used in version 2.1. The energy model is based on the description found on the Mfold website: http://www.bioinfo.rpi.edu/∼zukerm/rna/energy/node2.html, see also the Mathews et al. [1999] article. In the following the parameters used in version 2.0 of FOLDALIGN are used.

2.2.1 Hairpin-loops

A hairpin-loop is a stretch of unpaired nucleotides closed by a base pair. The base pair is not part of the loop. In Figure 2.6 the hairpin-loop nucleotides are colored blue and black. The score of an entire hairpin-loop has the following elements:

Single strand substitution Each nucleotide in the loop of the first sequence is aligned to a nucleotide or a gap in the second sequence. In the example there are five single strand substitutions: *G* ↔ *G*, *A* ↔ –, *C* ↔ *G*, *U* ↔ *U*, *A* ↔ *A*.

Loop lengths The lengths of the unpaired regions also have a cost. In the example the lengths are 5 and 4. The default minimum length is three nucleotides. The length cost for short loops is read from the loop length table.

End stacking The nucleotides at the ends of the loop (blue color) stack on top of the closing base pair (red color) if the length of both loops are longer than three nucleotides. The two stackings are: GA ⇔ *AU* and *GA* ⇔ GC. An extra non-*GC* cost has been added to the hairpin-loop stacking parameters. This cost is needed due to the way hydrogen bonds are counted in the nearest neighbor model [Xia et al., 1998]. If the base pairs *AU* and *GC* are not followed by other base pairs, then the *AU* and *GC* base pairs are not considered to base pair and are treated as part of the loop.

The score of the alignment in Figure 2.6 is:

$$\begin{aligned}
S_{hp} &= S_{ss-substitutions} + S_{lengths} + S_{hpstacks} \\
&= S_{ss}(G,G) + S_{ss}(A,G) + S_{ss}(C,-) + S_{ss}(U,U) + S_{ss}(A,A) + \\
&\quad S_{length}(5) + S_{length}(4) + S_{hpstack}(GA,AU) + S_{hpstack}(GA,GC) \\
&= 9 - 50 - 25 + 13 + 19 - 56 - 56 + 11 + 22 \\
&= -113
\end{aligned} \tag{2.1}$$

2.2.2 Stems

A stem is a series of base pairs stacked onto each other. The minimum length of a stem is two base pairs. A potential base pair without a neighboring base pair is not considered a base pair

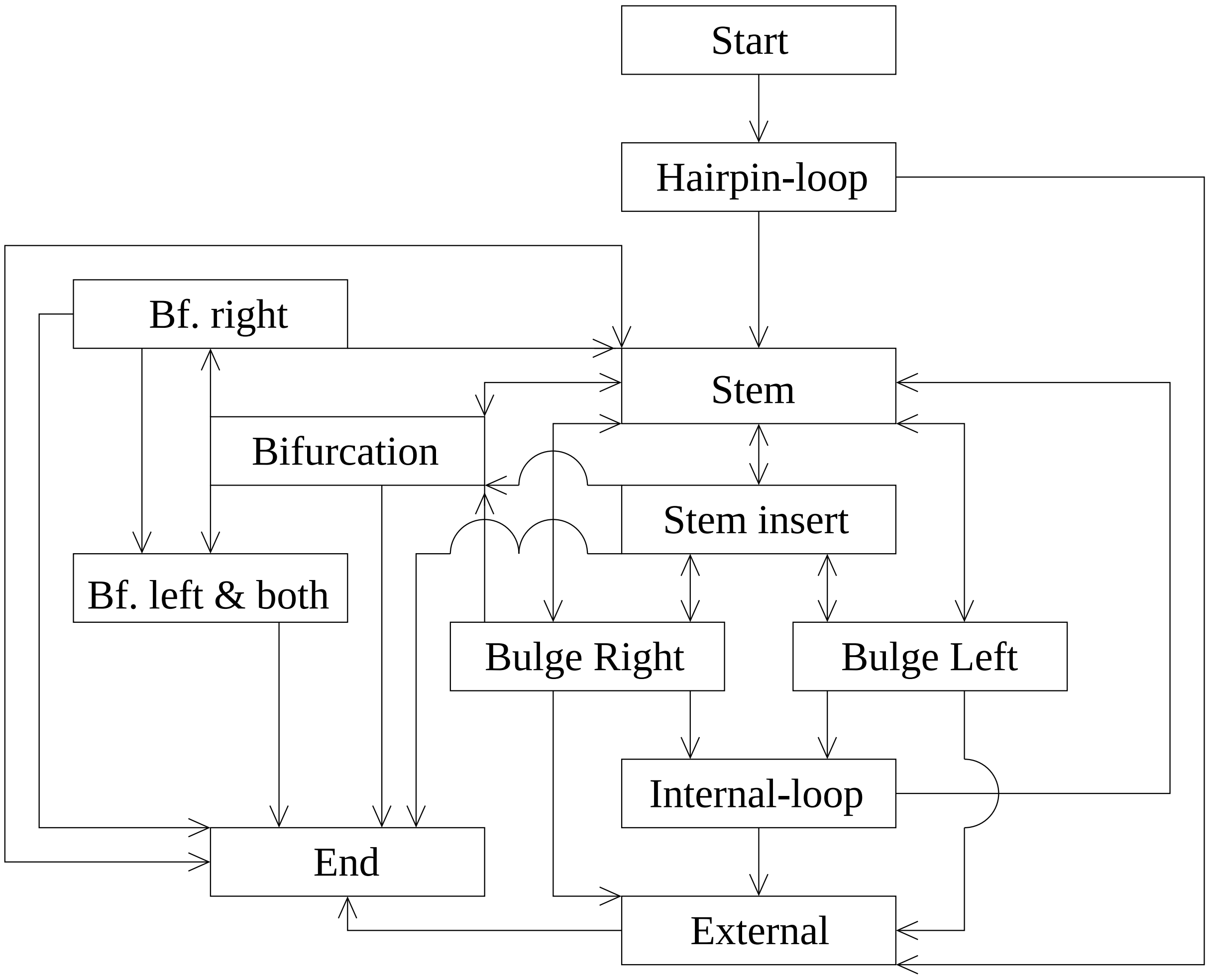


Figure 2.5: A simplified state chart for the 2.1 implementation of FOLDALIGN. The alignment always starts in the “Start” state which is a hairpin-loop state. The alignment ends in the “End” state. The “External” state recalculates the scores of the “Hairpin-”, “Bulge-”, and “Internal-” loop states to an “External” state score when needed. Unpaired nucleotides in the bifurcation states are scored in the same way as external states. The “Hairpin-loop” state aligns unpaired nucleotides in the hairpin context. The “Stem” state aligns base pairs in both sequences. The “Stem insert” state aligns a base pair in one of the sequences with two gaps in the other. “Bulge right” aligns bulges on the right side of a stem. “Bulge left” aligns bulges on the left side of a stem. The “Internal-loop” state aligns two internal-loops nucleotides. The “Bifurcation” state joins two substructures. The right structure must be in the “Stem” or “Stem insert” state. The state of the left structure must be: “Stem”, “Stem insert”, “Bifurcation”, “Bulge right”, or Bifurcation unpaired right (“Bf. right”). Bifurcation unpaired right aligns unpaired nucleotides on the right side of a branch point. Bifurcation unpaired left & both (“Bf. left & both”) aligns unpaired nucleotides on the left, right, and both sides of a branch point.

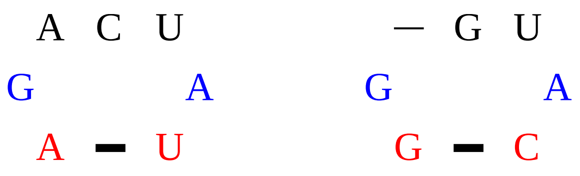


Figure 2.6: A hairpin-loop alignment. The base pairs (red) closing the loop are not part of the loop, but the score still depends on these nucleotides.

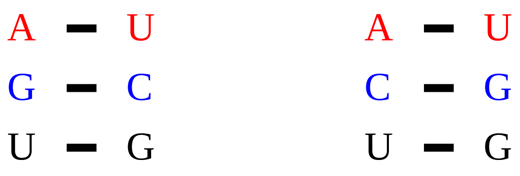


Figure 2.7: An alignment of two stems.

and is treated as part of a loop. Figure 2.7 shows an example of the alignment of two stem regions. In the 2.0 version of the algorithm a base pair can be inserted between two conserved base pairs. In version 2.1 a base pair can be inserted when the stem is at least one base pair long (the first base pair must be conserved), and the final stem must contain at least two conserved base pairs. The score of a stem alignment has two elements in both versions:

Base pair substitutions The cost of substituting the nucleotides of a base pair in one sequence with the nucleotides of the corresponding base pairs in the other sequence. Here it is: *AU* ↔ *AU*, *GC* ↔ *CG*, and *UG* ↔ *UG*.

Stacking The nucleotides of a base pair stack onto the nucleotides of its neighboring base pairs. In the example there are four stacks: *AU* ⇔ *GC* and *GC* ⇔ *UG* in the first sequence. *AU* ⇔ *CG* and *CG* ⇔ *UG* in the second sequence. The stem in the example is not complete. The red and black base pairs will in most cases also stack with their neighbors which are not shown in the figure.

The score of the alignment in Figure 2.7 is:

$$\begin{aligned} S_{stem} &= S_{bp-substitutions} + S_{stacks} \\ &= S_{bp}(AU, AU) + S_{bp}(GC, CG) + S_{bp}(UG, UG) + S_{stack}(AU, GC) + \\ &\quad S_{stack}(GC, UG) + S_{stack}(AU, CG) + S_{stack}(CG, UG) \\ &= 11 + 5 + 8 + 24 + 14 + 21 + 15 = 98 \end{aligned} \tag{2.2}$$

2.2.3 Bulge-loops

Bulges are a single strand region on one side of the molecule enclosed by base pairs, see Figure 2.8. The closing base pairs are not part of the bulge. The single strand region must be on the same side of the molecule in both sequences. The length of the bulge in one of the sequences may be zero if the length of the bulge in the other sequence is one or more. The score for a bulge alignment has these elements:

Single strand substitution The cost of substituting the single stranded nucleotides with each other. In the example there is a *C* ↔ *C* substitution.

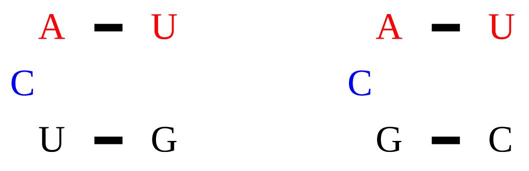


Figure 2.8: A bulge-loop section of an alignment.

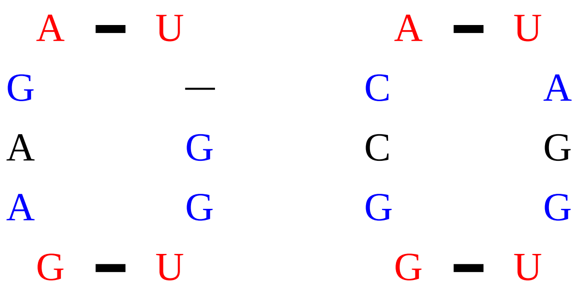


Figure 2.9: An internal-loop section of an alignment.

Length There is a cost for the length of the bulge. In the example both lengths are one.

Stacking If the lengths of the bulges in both sequences are zero or one, then the closing base pairs stack on to each other. This is the case in the example where the two stackings are: $AU \Leftrightarrow UG$ and $AU \Leftrightarrow GC$.

Non-*GC*-end cost This is an extra cost added each time a single stranded loop is closed by a non-GC base pair. The exception from this rule is the bulges where the closing base pair can stack on to each other i.e. the one or zero length bulges. In the example the cost is not added. If the bulges had been longer, the cost would have been: 3×-5.

The score of the bulge alignment in the figure is:

$$\begin{aligned} S_{bulge} &= S_{ss\_substitutions} + S_{lengths} + S_{stacking} + S_{non-GC} \\ &= S_{ss}(C,C) + 2 \times S_{length}(1) + S_{stack}(AU,UG) + S_{stack}(AU,GC) \\ &= 11 - 2 \times 38 + 10 + 24 = -31 \end{aligned} \tag{2.3}$$

2.2.4 Internal-loops

Internal-loops are single strand regions closed by two base pairs which are not bulge-loops, see Figure 2.9. This definition includes loops which in single sequence folding would be considered bulge-loops, but in the pairwise folding become internal-loop because they are on opposite sides of the surrounding stems. The closing base pairs in the figure are not part of the internal-loop.

The score of an internal-loop has several elements:

Single strand similarity The substitution cost for the nucleotides in the loops. In the example: $G \leftrightarrow C$, $A \leftrightarrow C$, $A \leftrightarrow G$, $- \leftrightarrow A$, $G \leftrightarrow G$, and $G \leftrightarrow G$,

Loop lengths A cost depending on the total length of the loop. In the example the total length of the loop in the first sequence is $\Delta_1 = 3 + 2 = 5$ and in the second sequence $\Delta_2 = 3 + 3 = 6$.

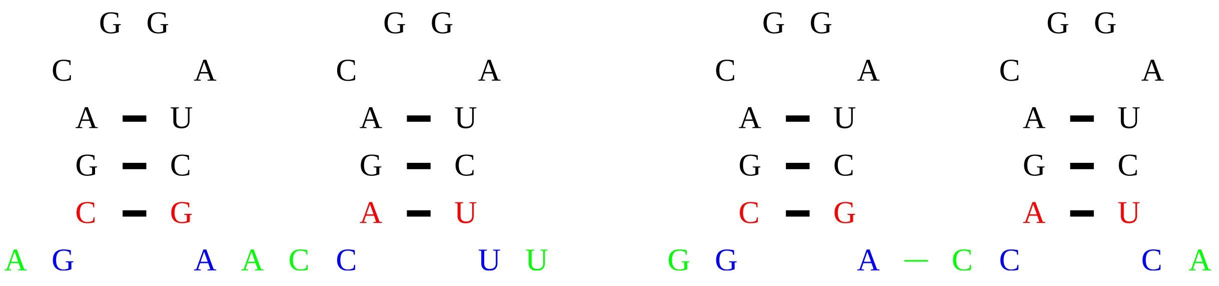


Figure 2.10: An alignment with a bifurcated-loop. The loop joins together two substructures. Only the single stranded nucleotides marked with blue and green colors are part of the loops. The example shows an open bifurcated-loop. The loop can also be closed by a stem. A closed loop is also called a multibranched loop.

Loop asymmetry There is a cost for loops of unequal lengths. The cost is directly proportional to the length difference between the two loops on each side of the molecule. In the example the differences are one for the left sequence and zero for the right sequence. The asymmetry cost is limited to a maximum value.

End stackings The unpaired nucleotides at the end of the loops stack with the closing base paired nucleotides. The non-*GC* base pair cost is assumed to have been added to the parameters already. In the example the four stackings are: *GG* ⇔ *AU*, *AG* ⇔ *GU*, *AC* ⇔ *AU*, and *GG* ⇔ *GU*.

The score of the internal loop alignment in the figure becomes:

$$\begin{aligned} S_{Internal-loop} &= S_{ss\ substitutions} + S_{lengths} + S_{asymmetry} + S_{Il\ end\ stackings} \\ &= S_{ss}(G,C) + S_{ss}(A,C) + S_{ss}(A,G) + S_{ss}(-,A) + S_{ss}(G,G) + \\ &\quad S_{ss}(G,G) + S_{length}(5) + S_{length}(6) + S_{asym}(1) + S_{asym}(0) + \\ &\quad S_{Il\ end\ stack}(GG,UA) + S_{Il\ end\ stack}(AG,GU) + \\ &\quad S_{Il\ end\ stack}(AC,UA) + S_{Il\ end\ stack}(GG,GU) \\ &= -25 - 22 - 18 - 50 + 9 + 9 - 18 - 20 - 5 + 0 - 7 + 4 - 7 - 7 \\ &= -157 \end{aligned} \tag{2.4}$$

2.2.5 Bifurcated-loops

Bifurcated-loops are loops of unpaired nucleotides between or on the outside of substructures, see Figure 2.10. In version 2.1 of the algorithm this context is also used as an external nucleotide context. Any single stranded nucleotide outside any base pair is scored using the bifurcation context. For an alignment without any base pair this context is also used. The score of the bifurcated-loop has these elements:

Single strand sequence similarity The cost for substituting the nucleotides in the two sequences with each other. In this example: $A \leftrightarrow G$, $G \leftrightarrow G$, $A \leftrightarrow A$, $A \leftrightarrow -$, $C \leftrightarrow C$, $C \leftrightarrow C$, $U \leftrightarrow C$, $U \leftrightarrow A$,

Number of unpaired nucleotides There is a cost added for each unpaired nucleotide in the loop. Often this cost is set to zero. Setting this cost and the single strand similarity costs to zero can lead to very long alignments since there is no cost for extending the bifurcation-loop.

Number of substructures A cost for adding new substructures to the loop. The first substructure is not counted. In the example there are two substructures. The cost is therefore added once.

Loop closing If there is a stem closing the two ends of the loop, a cost is added. This is not the case in the example, but the structure on the left would have been closed if the *AG* at the start of the sequence had base paired with the *UU* at the end.

Dangling ends (Only in version 2.0 of the algorithm). Unpaired nucleotides next to base paired nucleotides can stack onto the base pair. If an unpaired nucleotide can stack on more than one base pair, it only stacks on to the most favorable. Unfavorable stackings are ignored. The stacking cost are different for stackings to the $5'$ or the $3'$ side of the base pair. In the example there are the following stackings: *G* ⇔ *CG*, *C* ⇔ *AU*, *G* ⇔ *CG*, *C* ⇔ *AU*, *CG* ⇔ *A*, *AU* ⇔ *U*, *CG* ⇔ *A*, and *AU* ⇔ *C*.

Non-*GC* end base pair The extra cost added when the base pair ending a stem is not a *GC* pair.

The score of the example is:

$$\begin{aligned}
S_{bifurcated-loop} &= S_{ss\ substitutions} + S_{unpaired\ nt.} + S_{substructures} + \\
&\quad S_{end\ stackings} + S_{non-GC\ end} \\
&= S_{ss}(A,G) + S_{ss}(G,G) + S_{ss}(A,A) + S_{ss}(A,-) + \\
&\quad S_{ss}(C,C) + S_{ss}(C,C) + S_{ss}(U,C) + S_{ss}(U,A) + \\
&\quad S_{un.nt.}(8) + S_{un.nt.}(7) + S_{sub.str.}(1) + S_{sub.str.}(1) + \\
&\quad 2 \times S_{5'\ end\ stack}(G,CG) + 2 \times S_{5'\ end\ stack}(C,AU) + \\
&\quad 2 \times S_{3'\ end\ stack}(CG,A) + 2 \times S_{3'\ end\ stack}(AU,U) + \\
&\quad 2 \times S_{non-GC\ end} \\
&= -18 + 9 + 19 - 50 + 11 + 11 - 15 - 19 - 8 \times 0 - 7 \times 0 \\
&\quad -4 - 4 + 2 \times 0 + 2 \times 1 + 2 \times 11 + 2 \times 6 - 2 \times 5 \\
&= -34
\end{aligned} \tag{2.5}$$

2.3 Constraints

The FOLDALIGN algorithm is computational heavy. Without any constraints the algorithm has a time complexity of $O(L_I^3 L_K^3)$ and a memory complexity of $O(L_I^2 L_K^2)$. This means that if the lengths of the sequences are doubled, it will take roughly 64 times longer to make the alignment, and it will require 16 times as much memory. Using a twice as fast computer will only allow the alignment 12% longer sequences in the same amount of time.

To make the algorithm more useful constraints are used. Previous implementations of FOLDALIGN use three types of constraints [Gorodkin et al., 1997b, 2001b]:

The maximum motif length — λ The maximum motif length was limited to λ nucleotides.

The maximum length difference — δ The maximum length difference between two subsequences being aligned was limited to δ nucleotides.

Stem-loop Structures were limited to stem-loop structures, no bifurcations were allowed.

Using these constraints the time complexity was reduced to $O(L_I L_K \lambda\delta)$ and the memory complexity to $O(L_I L_K \lambda\delta)$. In the newer versions of FOLDALIGN the λ and δ constraints are still used while the stem-loop constraint is available as an option.

The λ constraint is now further exploited to lower the memory complexity to $O(\lambda^3\delta)$, at the cost of doubling the run time. This is done by splitting the shortest of the sequences into smaller (usually $2 \times \lambda$ nucleotides) chunks. These are then aligned to a λ nucleotides long window which is scanned along the other sequence. The method is explained in Havgaard et al. [2005b], in the section named ”Mutual scan of two sequences and the algorithmic complexity” on page 59. Figure 2 on the same page illustrates the concept.

The stem-loop constraint is very effectively lowering the time complexity with a factor of $O(\lambda\delta)$, but it also severely limits the biological problems for which the algorithm can be used. The constraint is therefore no longer used by default. To speed up the calculation of the bifurcations the context in which a bifurcation can occur, has been limited. A structure consisting of two substructures can usually be made from a wide range of substructures, see Figure 2.11. The bifurcated structure can be assembled from two substructures at all the points marked by lines. All ten possible combinations of substructures result in the same structure. Calculating only one of the combinations significantly speeds up the alignment. Exactly which combination that is used, depends on the implementation. The 2.0 version of FOLDALIGN uses the combination indicated by the red line. The bifurcation score is only calculated when the left substructure ends with a base pair, and the right structure is either stem, bifurcation, or bulge on the left side of the molecule. The 2.1 version uses the combination marked in blue. Here the bifurcation is only calculated when the start nucleotide of the left structure is base paired downstream, and the right side is a stem. The reason for the method change is that the 2.1 method exploits the constraint to also save large amounts of memory by only storing subalignments which end with base pairs for downstream positions in the I sequence.

When non-branched alignments are wanted, the memory requirement is only $O(\lambda^2\delta)$ during the scan phase since it is not necessary to use the long term memory (version 2.1 only). A linear

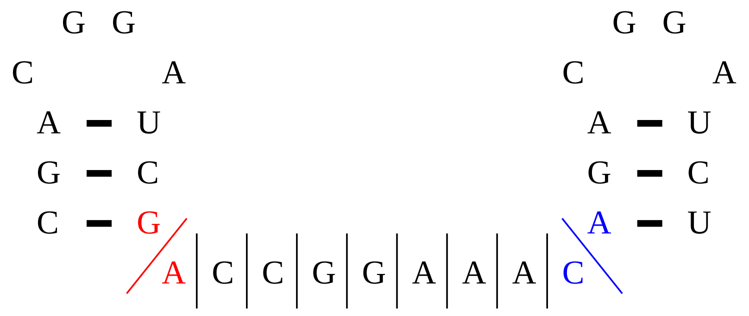


Figure 2.11: Bifurcation constraints. When two substructures are joined, there are usually many exactly similar ways to do it (The figure shows a single sequence example for clarity). In the figure ten slightly different pairs of substructures can all be joined into the same structure. FOLDALIGN therefore only calculates this structure once. In version 2.0 the pair of substructures separated by a red line is used. In 2.1 the substructure pair separated by the blue line is used. Limiting the bifurcation to the blue pair instead of the red pair allows the memory requirement to be lowered since it is only necessary to store subalignments which end with a base pair for the positions upstream of the current position.

space implementation is required before the backtrack stage can exploit this [Myers and Miller, 1988, Hirschberg, 1975].

2.3.1 Pruning

The dynamical pruning constraint added in version 2.1 requires that an alignment must have a minimum score depending on the length of the alignment. If the alignment score is below the minimum, then the alignment is not stored in the dynamic programming matrix. The recursion used in the algorithm has been rewritten so that it takes an alignment and expands it into new alignments. An empty cell cannot be expanded, and the algorithm can therefore move quickly to the next cell. This saves a lot of time and memory especially in the case where there is no motif in the sequences being aligned. The average time needed to align two 1000 nucleotide sequences using different values of λ can be seen in Figure (2.1).

Since gaps have relatively large negative scores, the use of pruning removes alignments with many gaps. While this is normally not a problem for local alignment where alignments with many gaps usually is not what is wanted, it is a problem for global alignment. For some global alignment cases FOLDALIGN does not report an alignment since all subalignments have been pruned away. To limit the number of these cases the pruning scheme is slightly adjusted for global alignment. Since the global alignment must reach from end to end of both sequences, the minimum number of gaps needed to make the alignment equals the length difference between the two sequences. The pruning score is made dependent not only on the length of the subalignment, but also on the length difference between the two subsequences. The global pruning score becomes:

$$\Theta_{global} = \Theta_{local}(l_I, l_K) - \min\{\text{abs}(l_I - l_K), \text{abs}(L_I - L_K)\} \times G_E \tag{2.6}$$

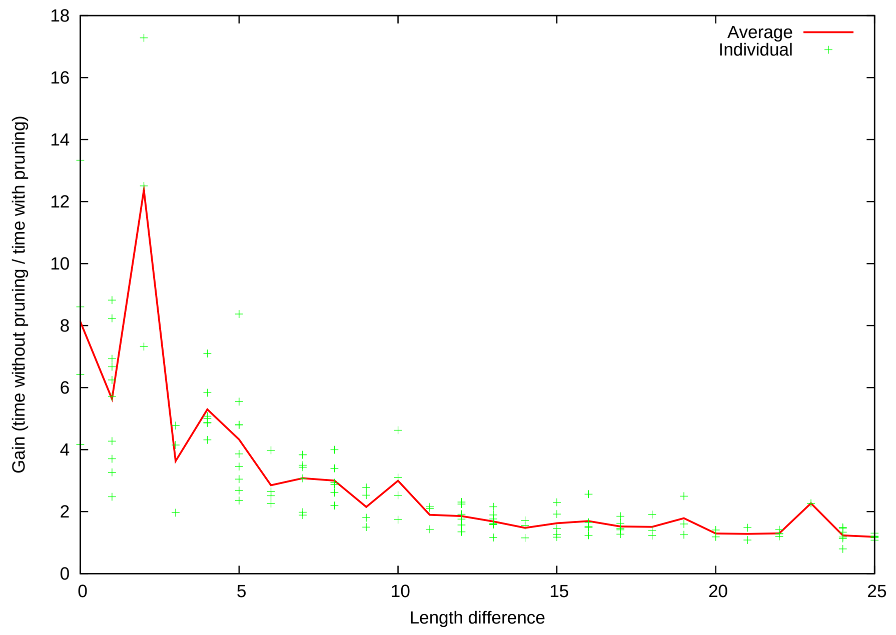


Figure 2.12: The curve shows the average speed gained by using pruning as a function of the length difference between the input sequences for global alignment. SRP sequences were used as input sequences. The maximum length difference (δ) was 25.

Where Θ_{global} and Θ_{local} are the pruning scores in the global and local pruning scores. $l_I = j - i + 1$, $l_K = l - k + 1$, and G_E is the gap elongation cost. In this way sequences of widely different lengths can usually be globally aligned, but at the cost of slowing down the algorithm. Figure 2.12 shows the speed gain as a function of length difference between the input sequences. At length difference 25 the speed gain has dropped to an average of 1.2. Further work is needed to make pruning work efficiently for large length differences.

2.4 Parameters

The most commonly used method for RNA folding is that of energy minimization [Zuker, 2003, Hofacker, 2003]. The parameters used in energy minimization are based on thermodynamical studies of small oligoes [Mathews et al., 2004, 1999, Tinoco Jr. et al., 1973]. In sequence alignment the most commonly used parameters are log-odds scores [Henikoff and Henikoff, 1992, Altschul, 1991, Altschul et al., 1997]. FOLDALIGN uses a combination of the two types of parameters. The substitution matrices are log-odds scores, and the structural parameters are taken from energy minimization.

2.4.1 Energy

The minimum free energy of an RNA structure is calculated using the nearest neighbor model. This builds on the assumption that the main contribution to the free energy of an RNA structure is the stacking of neighboring base pairs on to each other [DeVoe and Tinoco Jr., 1962]. The simple model has been expanded into a complex model based on the stacking of nucleotides, the lengths of unpaired loops, and ad-hoc rules [Mathews et al., 1999, 2004]. The parameters are determined by experiments which study the thermodynamics of melting and folding of small oligos, and by fitting parameters estimated from multiple alignments to experimental results. The energy parameters used in FOLDALIGN were taken from the Mfold package [Mathews et al., 1999] (with a few minor changes in Havgaard et al. [2007]).

2.4.2 Ribosum-Like

The different parts of an RNA molecule can evolve at different speeds. The nucleotides at some positions are essential for the function of the molecule and rarely change. The nucleotides at other positions are parts of stems and just need to base pair with the nucleotides at other specific positions. These nucleotides can change more frequently by compensating mutations. Nucleotides at other positions might serve as spacers, and the specific type of nucleotide is of little importance which allows for frequent changes.

The substitution matrices used are based on the RIBOSUM matrices [Klein and Eddy, 2003]. The RIBOSUM matrices are based on the ideas behind the BLOSUM matrices used in protein alignment [Henikoff and Henikoff, 1992], but have separate sub-matrices for single strand regions and base paired regions.

Some substitutions are more likely to be seen between RNA molecules than between random sequences, and vise versa. A substitution cost is therefore calculated as the log-odds ratio between the probability of the substitution between RNA molecules (P') and the probability of the substitution between random sequences (P) [Altschul, 1991]. The substitution cost for substituting the nucleotides n_i and n_k with each other is:

$$S(n_i, n_k) = \log_2 \frac{P'(n_i, n_k)}{P(n_i, n_k)} = \log_2 \frac{P'(n_i, n_k)}{P(n_i)P(n_k)} \tag{2.7}$$

In the second step the nucleotides at positions i and k in the random RNA are assumed to be independent of each other and of any other nucleotide. The substitution probability is therefore estimated to be the product of the background probabilities, $P(n_i)$ and $P(n_k)$, for the two nucleotides. For base paired nucleotides this becomes:

$$S(n_in_j, n_kn_l) = \log_2 \frac{P'(n_in_j, n_kn_l)}{P(n_in_j, n_kn_l)} = \log_2 \frac{P'(n_in_j, n_kn_l)}{P(n_i)P(n_j)P(n_k)P(n_l)} \tag{2.8}$$

where n_i & n_j base pair, and n_k & n_l base pair. The probabilities are estimated with frequencies counted in multiple alignments.

The substitution matrices can be optimized to different evolutionary distances. In the BLOSUM based RIBOSUM-Like scheme this is done by clustering the sequences of the multiple alignment according to sequence identities before the frequencies are counted. For the clustering a sequence identity cutoff is selected. If a sequence is more identical to any one sequence already in a cluster than the cutoff, then the sequence is put into that cluster. For each cluster substitutions between the sequences in the cluster and the sequences in all the other clusters are counted. This is similar to what is done for the BLOSUM matrices, but unlike the RIBOSUM matrices where substitutions between sequences in the same cluster also are counted [Klein and Eddy, 2003]. Each count is weighted by the geometric mean of the cluster sizes.

The RIBOSUM-Like matrices are made from a multiple alignment of the 1995 version of the ribosomal Small Subunit (SSU) database [Van de Peer et al., 1994]. The cleaned alignment was supplied by Robert J. Klein [Klein and Eddy, 2003 and personal communication]. The alignment had been cleaned by removing sequences with more than 5% ambiguous nucleotides, or sequences where more than 50% of the base paired positions were missing. The resulting alignment has 2492 sequences.

Figure 2.13 shows the number of clusters as a function of the clustering cutoff. The number of sequences per cluster for some of the cutoffs can be seen in Table 2.2. Figure 2.14 shows the number of sequence pairs being compared. In Henikoff and Henikoff [1992] substitutions are counted in blocks of ungapped multiple alignments. In the Ribosum-Like matrices the sequences have gaps. The gaps are counted as mismatches during the calculation of sequence identities. This changes the clustering slightly. In Figure 2.13 and 2.14 two curves are shown. One where gaps are counted as mismatches, and one where gaps are not counted. From the Figures and the numbers in Table 2.2 it is clear that using more families is likely to improve the matrices, especially for low identities as the numbers of clusters are very low.

The “Gaps not counted” curve and the numbers in Table 2.2 show that there is little data below the 80% cutoff and very little data below the 70% cutoff. Using multiple alignments of other RNA families would be the best way to improve these numbers.

2.4.3 Combining energy and substitution parameters

The energy and substitution parameters have to be combined into one score-matrix. This was done using a simple trial and error approach. In Havgaard et al. [2005b] the two RIBOSUM-Like matrices are scaled independently. The performance is measured for different values of the weights, and the optimal weights are selected. Together with the clustering percentage this gives the score-matrix three parameters which have to be optimized. It was found that the single strand substitution matrix should have a larger weight than the base pairing matrix. It

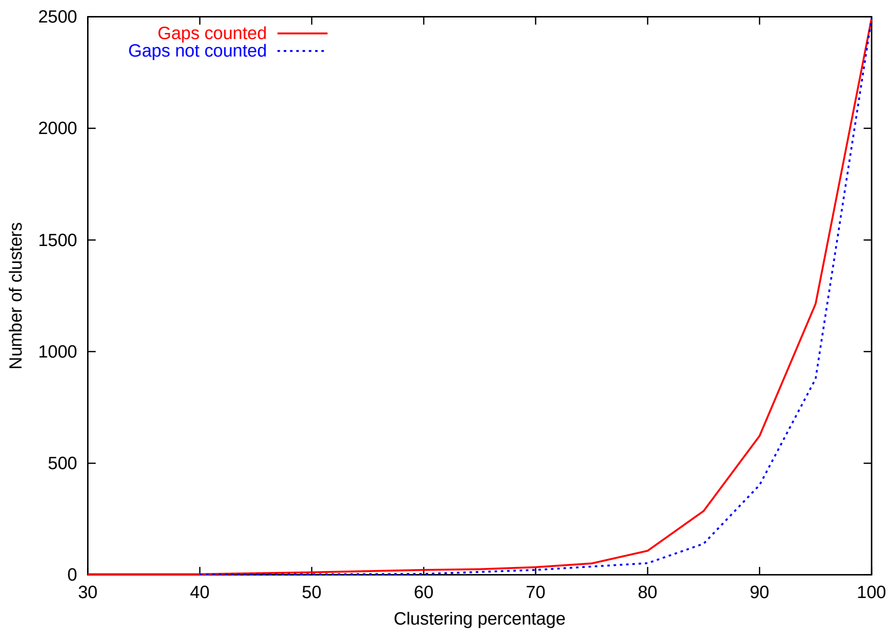


Figure 2.13: The number of clusters as a function of the clustering cutoff. A sequence is put into a cluster if the sequence similarity between the sequence and any sequence in the cluster is above the cutoff.

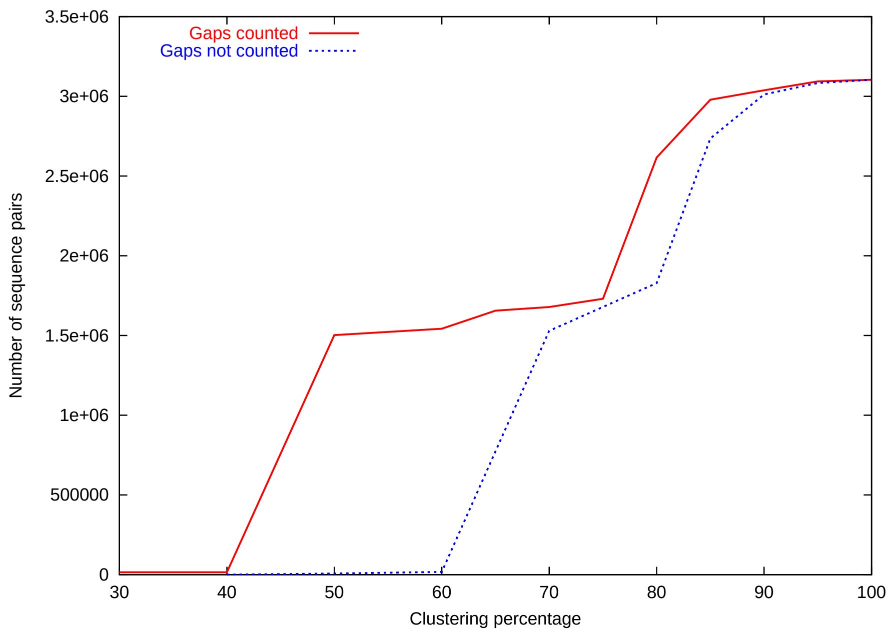


Figure 2.14: The number of sequences compared as a function of the clustering cutoff. The sequences from one cluster are compared to all the sequences in another cluster. The counts from each cluster/cluster comparison are normalized to one.

50%		60%		70%		80%	
Size	#	Size	#	Size	#	Size	#
3	1	1	1	1	7	1	19
2489	1	3	2	2	6	2	9
		2485	1	3	3	3	5
				6	2	4	4
				11	1	5	1
				22	1	6	2
				874	1	8	1
				1545	1	11	4
						17	1
						21	1
						44	1
						50	1
						51	1
						782	1
						1390	1

Table 2.2: The size is the number of sequences in a cluster in the "Gaps not counted" case, see Figures (2.13) and (2.14). # is the number of clusters with this size. The numbers 50% – 80% are the clustering percentages. If the sequences are clustered with 60% identity, then there will be four clusters: One with one sequence, two with three sequences, and the final cluster will have 2485 sequences.

has been reported that even though this set of parameters makes good sequence alignments it has a low base pair sensitivity when the aligned sequences have moderate or high sequence identities [Dowell and Eddy, 2006]. A reasonable explanation for this is that the high single strand substitution cost makes it very favorable for a nucleotide to be single stranded when the sequence similarity is high. To avoid this problem the substitution matrices were scaled using equal weights in Havgaard et al. [2007].

In addition to the clustering percentage and the matrix weights gap penalties are also needed. Affine gaps are used. There are two gap penalties: One for initiating a new gap and one for elongating an already opened gap. In general the elongation gap penalty was fixed to half the gap initiation penalty. The algorithm's performance for a given set of gap penalties is dependent on the problem for which it is used. Separating biologically relevant alignments from spurious alignments demand one set of penalties. Predicting the correct structure of a given RNA family another set. For this reason FOLDALIGN has two build-in score matrices: one for local alignment and one for global alignment. The optimal gap penalties are often different for different RNA families, see Havgaard et al. [2005a] supplementary material Figure S1 on page 74, Havgaard et al. [2005b], and Mathews and Turner [2002].

2.5 Alignment selection

The main use for FOLDALIGN 2 is to scan a pair of sequences of locally conserved structures. As the alignments are local, there might be more than one conserved biologically relevant structure in the input sequences. Therefore a script, named locateHits, which can locate non-overlapping alignments and evaluate their significance, has been included in the software package.

The input into the script is a list of alignment coordinates and scores. This list is produced by FOLDALIGN when option *-plot_score* is used. In version 2.0 the program outputs the coordinates and the score of the highest scoring alignment starting at all pairs of the (i, k) coordinates in the two sequences. Version 2.1 only outputs the information when the alignment is a structural alignment. Such an alignment must have at least two base pairs. The score of random alignment is expected to grow linearly with the logarithm of the alignment length [Chvátal and Sankoff, 1975]. For the pair of coordinates (i, k) the best scoring alignment, $S_{LS}(i, k)$, is therefore in version 2.1 defined as the one with the highest score compared to the logarithm of the longest of the two subsequences, i.e.

$$S_{LS}(i,k) = \max_{j,l} \left(\frac{S(i,j,k,l)}{\log_2(\max(j-i+1, l-k+1))} \right) \tag{2.9}$$

2.5.1 Non-overlapping alignments

One reasonable choice of alignments is the highest scoring which do not overlap each other in both sequences. Havgaard et al. [2005b] describes an algorithm aimed at finding these alignments (page 60, section "Selection of a hit region"). Step two of the algorithm is incorrectly described. The correct algorithm is:

1. Find the best scoring alignment $D_{ij,kl}$.
2. Remove all alignments $D_{i'j',k'l'}$ for which
   $i' \leq j$ and $j' \geq i$
   and
   $k' \leq l$ and $l' \geq k$.
3. Repeat until all alignments have been removed, or a predetermined threshold is reached.

The form described here is the one used to produce the results in the paper and implemented in the locateHits script. If the incorrect algorithm had been used, an alignment $D_{i'j',k'l'}$ with $D_{i'j',k'l'} < D_{ij,kl}$ where $i' < i, j' > j, k' < k, l' > l$ would not have been removed.

An algorithm like the "declumbing" algorithm [Waterman and Vingron, 1994] which removes all alignments which share common aligned nucleotides n_i and n_k, could also be used in theory, but in reality it would be computationally too heavy since it requires backtracking of the entire dynamic programming matrix. The "island" algorithm would also be computational infeasible if at all possible [Olsen et al., 1999]. For sequence similarity based methods like BLAST and FASTA a speedy implementation of algorithms like "declumbing" and "island" is important because they have the same time complexity as the full algorithm. The long runtime for FOLDALIGN makes it less important how fast the hit location algorithm is as long as it is much smaller than that of FOLDALIGN.

Figure 2.15 shows the ten first alignments from the locateHits 2.0 script used on a real sequence pair. Shuffled sequences were used to make Figure 2.16. The coordinates are in the *i, j, k,* and *l* columns. The alignment score is in the *Score* column. *Z* is the Z-score. The *P* column will be discussed in the next section. *Rank* is the position in the list sorted by scores. The *Rank* is used in the N-best alignments selection scheme in Havgaard et al. [2005b]. The first three lines from the real alignment figure show the location of three tRNA genes. The alignments overlap in the *AC069454* sequence, but not in the *V00158* sequence.

2.5.2 Significance of an alignment

The second part of the problem of finding the correct alignment, namely determining which alignments are due to conserved sequence/structure, and which are just random alignments, is addressed in this section.

It has been shown that for scoring algorithms like FOLDALIGN it is reasonable to expect the alignment scores for alignment of random sequences to be extreme value distributed [Karlin and Altschul, 1990, Heyer, 2000, Gumbel, 1958]. If the distribution is known, it is possible to calculate the probability that an alignment with a given score or better would be found by chance.

For fast sequence similarity alignment methods like BLAST and FASTA it is possible to align many random sequences to estimate the parameters of the extreme value distribution [Altschul et al., 1990, Pearson and Lipman, 1988]. For a relatively slow method like FOLDALIGN aligning a comparable number of random sequences will take much longer. One way to circumvent this problem is to use the observation by Waterman and Vingron [1994] and Olsen et al. [1999] that a single alignment of two sequences can contain multiple independent local alignments.

The extreme value distribution has two parameters, Λ[1] and κ, which are depended on the parameters of the scoring scheme and the sequences being aligned. Λ and κ are estimated using the method described in Altschul et al. [2001]. The main points are repeated here: The number of subalignments in a random alignment that will have a score better than a score x, is taken to be Poisson distributed. The probability that an alignment with a score $S' \geq x$ will be produced by chance, is estimated by:

$$P(S' \geq x) \approx 1 - \exp(-\kappa L_I L_K \exp^{-\Lambda x}) \tag{2.10}$$

The estimate is expected to be better for higher alignment scores. The parameters are therefore estimated using only scores above a threshold c. The maximum likelihood estimate for Λ is:

$$\Lambda = \ln\left(1 + \frac{1}{\frac{1}{N}\sum_{i=1}^{N}(S(i) - c)}\right) \tag{2.11}$$

[1] Λ is usually named λ in other texts, but has been renamed here to avoid confusion with maximum motif length λ

Name	i	j	Name	k	l	Score	Z	P	Rank
V00158	173	245	AC069454	297	367	632	8.64	0.001	1
V00158	12	82	AC069454	297	367	602	8.28	0.002	2
V00158	89	159	AC069454	297	367	561	7.78	0.004	3
V00158	113	131	AC069454	183	203	192	3.35	0.981	4
V00158	202	245	AC069454	183	226	175	3.15	0.996	5
V00158	431	445	AC069454	324	338	158	2.95	0.999	6
V00158	33	62	AC069454	418	447	126	2.56	1.000	7
V00158	366	384	AC069454	184	202	125	2.55	1.000	8
V00158	464	480	AC069454	184	202	124	2.54	1.000	9
V00158	434	499	AC069454	405	469	116	2.44	1.000	10

Figure 2.15: The ten best local alignments between two 500 nt. long sequences. The first six columns are sequence names and start and end positions. The score column is the FOLDALIGN score, Z is the Z-score, P the P-value, and Rank is the position in the list. The parameters for the Z-value distribution are $\mu = -87.4,\ \sigma = 83.3$. The parameters for the P-score extreme value distribution are $\Lambda = 0.0189,\ \kappa = 0.000604$. The cutoff score was 10, 125 alignments were used, and the sum of the scores was 7792.

Name	i	j	Name	k	l	Score	Z	P	Rank
V00158	206	336	AC069454	137	261	176	8.93	0.313	1
V00158	370	404	AC069454	432	467	138	7.73	0.793	2
V00158	323	421	AC069454	266	355	131	7.51	0.871	3
V00158	120	170	AC069454	433	485	130	7.47	0.881	4
V00158	190	201	AC069454	36	47	123	7.25	0.937	5
V00158	170	200	AC069454	320	349	114	6.97	0.980	6
V00158	52	132	AC069454	283	368	112	6.91	0.985	7
V00158	80	113	AC069454	424	459	102	6.59	0.998	8
V00158	426	466	AC069454	186	226	99	6.50	0.999	9
V00158	4	16	AC069454	301	313	93	6.31	1.000	10

Figure 2.16: The ten best local alignments between the two sequences shuffled. The parameters for the Z-value distribution are $\mu = -107.0,\ \sigma = 31.7$. The parameters for the P-score extreme value distribution is $\Lambda = 0.0377,\ K = 0.00115$. The cutoff score was 150, one alignment score was used, and the score was 176. Clearly this estimate of the distribution parameters makes no sense.

Here $S(i)$ is the score of a non-overlapping random alignment with a score better than c, and the sum is taken over these alignments. N is the number of alignments with scores better than c. For κ the maximum likelihood estimate is:

$$\kappa = \frac{N \exp^{\Lambda c}}{L_I L_K} \tag{2.12}$$

Altschul et al. [2001] does not specify a method for selecting the cutoff value c. Version 2.0 of the locateHits script is supposed to calculate several values of Λ and select the smallest c for which Λ is smaller than the Λ for the next c value. This choice was inspired by Figure 3 in Altschul et al. [2001] where the value of Λ decreases until it has almost reached the best estimate after which it starts to fluctuate. Unfortunately there is a bug in the script, and the c value chosen is the smallest c for which Λ is larger than the Λ for the next c value. The script therefore usually picks a c value lower than it should, but this is not a big problem since a low c value usually yields good results. A problem with the "try several values of c" method is that it often picks a c value for which there is only one alignment with a better score. See Figures (2.16) and (2.17).

In version 2.1 of the locateHits script the problem of one value distribution estimates is fixed by always using $c = 0$. This cutoff was selected since a score of 0 is not negative and usually requires some structure in the alignment. The distribution of the number of hits used to find the parameters can be seen in Figure (2.18).

With a relatively slow method like FOLDALIGN making a large number of random alignments is not always possible. One solution to this problem is to use the non-overlapping local alignments from a non-random alignment. When two long sequences are aligned, then most of the non-overlapping local alignments will usually be spurious alignments. From these alignments a very rough estimate of the parameters can be made. Figure (2.19) shows the distributions of P-values for real and shuffled data. The small peak in the 0.3 to 0.4 range of the P-value distribution for the shuffled sequences is due to the distributions estimated from one value.

The biggest problem with this approach is that any non-random alignment will bias the estimate. In the 2.0 version of the locateHits script all alignments are used. But using a fixed cutoff in version 2.1 makes it possible to use an iterative method to remove significant alignments from the estimate. The iterative scheme is:

1. Estimate Λ and κ using the available scores.
2. Calculate the significance of each of the alignments. If an alignment is found to have a significant score, remove it.
3. If a significant alignment was found in step 2, go to step 1.

Using this method the estimate of Λ and κ is no longer biased by any real alignments found (those not found will still bias the estimate). Unfortunately this method introduces another bias by removing the high scoring spurious alignments. The iterative method cannot be used together with the method for selecting the cutoff c as the combination of the two methods very often leads to the one value distribution estimates problem. An example of the effect of removing the significant alignments from the estimate can be seen in Figure (2.20).

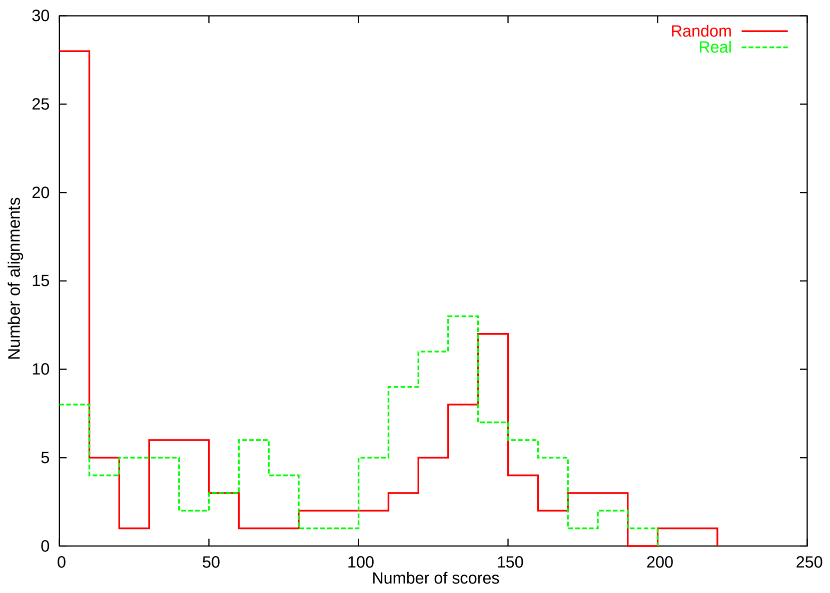


Figure 2.17: The extreme value distribution parameters were estimated for each of 99 500 nt. against 500 nt. alignments. The x-axis shows the number of scores used to estimate the parameters. The y-axis shows the number of alignments for which a given number of scores was used. Version 2.0 of the locateHits script was used. For about two thirds of the real alignments the method uses the scores of more then 50 non-overlapping alignments. For shuffled sequences the method is less useful with more than half of the alignments having less than 50 useful scores.

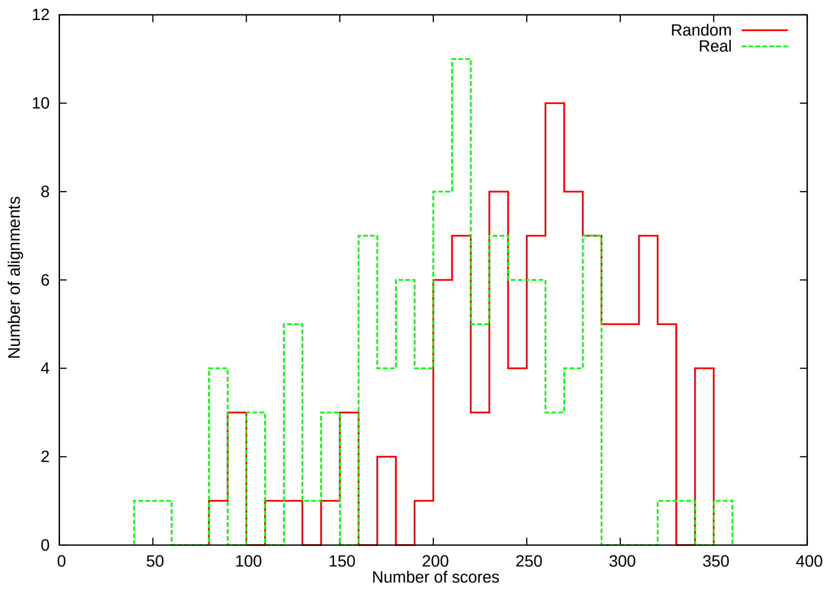


Figure 2.18: The “Number of scores” is the number of scores used to estimate the extreme value distribution. The “Number of alignments” is the number of alignments for which the extreme value distribution parameters were estimated using the number of scores. See also Figure (2.17). Version 2.1 of the locateHits script was used. The number of scores used to estimate the parameters is less critical.

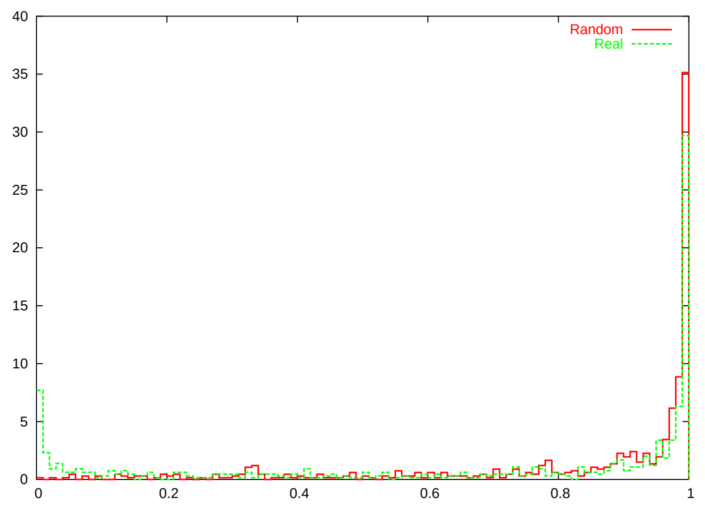


Figure 2.19: The p-value distribution for real and shuffled sequences (V. 2.0). Only p-values less than one were included. For the real sequences there is a clear peak at the small p-value. It is clear that the real sequences contain many more high scoring alignments than the shuffled alignments. For the alignment of shuffled sequences there is a very small peak between 0.3 and 0.4. This peak is most likely due to the high number of alignments for which only one score was used to estimate the distribution parameters, see Figure 2.17.

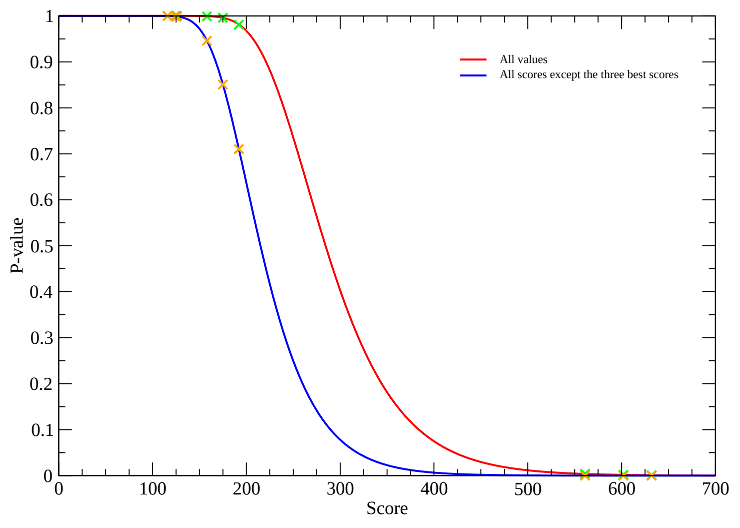


Figure 2.20: An example of the effect of the correct alignments on the estimates of the p-values. The curves show the p-value estimates as functions of score. The x's are the alignment scores and their estimated p-values. Including the correct alignment scores in the estimate leads to significantly different p-values for three of the alignments. In this case the bias gives a better separation between annotated and unannotated structures. In other cases it leads to false negatives.

While the method rapidly provides an estimate of the probability, the estimate can be biased in a number of ways:

True alignments The alignments used are not random alignments. Assuming that correct alignments score better than a random alignment, any correct alignment will bias the estimate. This will raise the estimate for a given score and thereby raise the number of false negatives, see Figure 2.20. This is a bigger problem for version 2.0 of the script than for the 2.1 version.

Missing high scoring spurious alignments In 2.1 the alignments found to be significant are removed. This will also remove any high scoring spurious alignments.

Cutoff The methods used to find the cutoff (try several or fixed $c = 0$) are not likely to be the correct/optimal methods.

Number of alignments In some cases the parameters are estimated from an extremely low number of alignments. In the worst case the distribution is estimated from the score of one alignment. This is mainly a problem for the 2.0 version.

Edge effects The estimated P-value is biased by the finite length of the sequences. This effect could be removed by using only alignments which are more than λ nucleotides from the end of the sequences. This is not done since the effect is expected to be small compared to some of the other biases.

Short alignment effects For short alignments it is impossible for the alignment to contain bifurcations. Also the low number of gaps bias described in Altschul et al. [2001] takes effect here.

Using parameters estimated from shuffled sequences removes the **True alignments** and **Missing high scoring spurious alignments** bias' and is highly recommended.

2.6 Data

2.6.1 Local alignment

The data used to tune and test FOLDALIGN version 2.0 should test the new features of this version. At the same time it should also demonstrate that FOLDALIGN can be used to find biologically relevant results which cannot be found by other existing methods. The following three main criteria for selecting structures/sequences were used:

Structure The new version can align bifurcated structures. The RNA structures used should therefore be bifurcated structures.

Low similarity Sequence identity is the most commonly used tool to locate new ncRNA genes. Using sequence pairs with pairwise identity below 40% show that FOLDALIGN can align sequences with low similarity.

Energetically indistinguishable Some RNA sequences have folding energies which are significantly lower than that of the surrounding sequence. The algorithm should work in cases where this is not true. To ensure this any sequence pair where at least one of the sequences has a free folding energy significantly lower than the surrounding sequence, were discarded.

Due the high memory complexity of version 2.0 the length of the RNA structures were limited to a maximum of 150 nucleotides. To keep any one sequence from dominating the dataset, the number of pairs a sequence could be a part of, was limited to three pairs, except for tRNA sequences which were only allowed to be in one pair. All multiple alignments were redundancy reduced to a maximum of 90% similarity before use. See Havgaard et al. [2005b] page 61 for details.

The local dataset contains 99 sequence pairs: 2 5S rRNA pairs, 5 Purine pairs, 21 THI pairs, 65 tRNA pairs, and 6 U1 pairs [Szymanski et al., 2002, Sprinzl et al., 1998, Zwieb, 1996, Griffiths Jones et al., 2003]. There are two versions of the data set: The structure version contains just the sequence where the structure is. The localization version contains 500 nucleotide long sequences where the sequence of the structure is located at a random position. The surrounding contexts were taken from GenBank [Benson et al., 2004]. The context sequence contains other RNA genes in some cases. The localization version of the dataset therefore contains 295 complete RNA genes. Partial genes are not annotated.

In addition to the local dataset a local SRP dataset was also made. This dataset contains eight sequence pairs. The sequences in the localization version are 1000 nucleotides long. The SRP genes are ~ 300 nucleotides long. For details see Havgaard et al. [2005b] page 61.

2.6.2 Global alignment 2.0

In relation to the web server a score matrix optimized for global alignment was needed. For a global dataset there is no need for a surrounding context. Sequences which do not have a context, can therefore be used. Therefore a global alignment dataset was made. It contains sequences from the 5S rRNA, tRNA, and U1 databases [Szymanski et al., 2002, Sprinzl et al., 1998, Zwieb, 1996]. The dataset is described in the supplementary material of the web server [Havgaard et al., 2005a], see page 72 for details. This dataset also comes in flavors: One where all pairs are less than 40% similar, and one where the maximum similarity is 70%.

An extra SRP dataset was also made for the global set. Details about this data set are also in the web server supplementary material [Havgaard et al., 2005a], see page 72.

2.6.3 Global alignment 2.1

The pruning constraint and better memory utilization allow the newest version of FOLDALIGN to align more and longer sequences than previous. A new global dataset was therefore made. Sequences were taken from the 5S rRNA, RNaseP, and tRNA databases [Szymanski et al., 2002, Brown, 1999, Harris et al., 2001, Sprinzl and Vassilenko, 2005]. Sequence pairs of up to 90% identity were included. The 5S rRNA and tRNA parts of the dataset were used to select the score matrix parameters. The RNaseP and SRP parts of the dataset were used to evaluate the performance. The dataset is further described in the supplementary material of Havgaard et al. [2007], see page 86.

The dataset made by Dowell and Eddy [2006] is also used in the supplementary material of Havgaard et al. [2007]. This dataset contains sequences from the seed alignments of 5S rRNA and tRNA from RFAM [Griffiths Jones et al., 2005].

Chapter 3

Results and conclusion

3.1 Results

The following six papers are the main results of this Ph.D. project:

Havgaard et al. [2005b] This paper introduces version 2 of the FOLDALIGN algorithm. The new version uses a minimum free energy model in combination with substitution scores to locally align the sequences. The other major change affecting the biological properties of the alignment is the addition of the possibility for bifurcated loops. The main algorithmic improvements are the use of the λ parameter to limit the memory consumption from $O(L_I L_K \lambda\delta)$ to $O(\lambda^3\delta)$. Allowing for bifurcated structures makes the algorithm significantly slower. To lower the run time the number of times the same bifurcated structure is calculated, is limited to one.

Havgaard et al. [2005a] The main result in this paper is the FOLDALIGN web-server — `http://foldalign.ku.dk`. In the supplementary material results for global alignment are presented.

Havgaard et al. [2007] This paper introduces the pruning constraint which efficiently limits the time and memory requirements of the algorithm without sacrificing the comparative information. The limit on the number of times a given bifurcated structure is calculated, is used to reduce the memory usage. A “divide and conquer” algorithm is introduced to try to keep the memory usage low during global alignment and backtrack. The two major changes to the energy model are that external unpaired nucleotides are now always treated in the same way as unpaired nucleotides in multibranched loops, and that more insert base pairs are allowed. Global alignment is treated in the supplementary material. Dowell and Eddy [2006] shows that while FOLDALIGN 2.0 makes good global alignments the base pair sensitivity is below that of comparable methods. Here it is shown using the same dataset that the new energy model fixes this problem. The improvement most likely stems from the better handling of external unpaired nucleotides.

Torarinsson et al. [2007] This paper describes FOLDALIGNM. It is a reimplementation and improvement of the PMcomp algorithm [Hofacker et al., 2004]. The algorithm makes one or more multiple alignments. To make the multiple alignment it aligns base pairing probability matrices. These can either be made using the McCaskill algorithm [McCaskill, 1990] or FOLDALIGN. To make more than one multiple alignment from a collection of sequences the sequences are first clustered by their pairwise FOLDALIGN score. Then a multiple alignment is made for each cluster.

Torarinsson et al. [2006] In this paper sequences from 10 human chromosomes were aligned to corresponding mouse sequences. The sequences that were aligned, are not part of the alignments made using traditional sequence alignments, but the sequences adjacent to them in one end were. In the other end there were gaps, repeat sequences, or a new alignment. Of the 73940 sequence pairs aligned 1297 pair yielded a significant alignment. The false positive rate was estimated to be $\sim 50\%$. Approximately 3600 significant alignments would have been found if all chromosomes had been used. 36 top candidates were selected for testing in the laboratory. These were tested in mouse tissue using PCR, and 32 were found to be expressed. Twelve of these 32 candidates were selected for further testing using Northern blots. Expression was confirmed for four of them. A database containing the significant candidates is available at `http://genome.ku.dk/resources/hm_ncrna_scan`.

Gorodkin et al. [2006] In the paper sequence logos [Schneider and Stephens, 1990, Gorodkin et al., 1997a] and Average Log Likelihood Ratios (ALLR) scores [Wang and Stormo, 2003] are used to show that the sequence motif of microRNAs from the left side of the precursor stem is different from the motif of those from the right side of the precursor.

3.2 Conclusion

FOLDALIGN is a tool for aligning RNA sequences using their structure and sequence similarities. It can make both local and global alignment. A tool for evaluating the significance of local alignments is included. The papers Havgaard et al. [2007, 2005a,b] introduces several improvements to the core algorithm. The improvements include a more biologically relevant energy model, including bifurcated structures. Time and memory requirements have been significantly lowered by the use of dynamic pruning. The scanning scheme, which splits the sequences into smaller chunks, limits the memory requirements even further. It is now possible to align 500 nucleotides long sequences on ordinary hardware in reasonable time both globally and locally. This is significant because many ncRNAs have lengths below 500 nucleotides [Huttenhofer et al., 2005]. FOLDALIGN is a user friendly program which combines good performance with low resource requirements (for a Sankoff algorithm).

According to Gardner et al. [2005] (Figure 3B) FOLDALIGN 2.0 is the best available algorithm for pairwise global alignment of tRNA sequences. Dowell and Eddy [2006] also finds that FOLDALIGN 2.0 is very good at making global alignments. But they also find that the method is not very sensitive for detecting base pairs compared to other methods. Figure S2 in the supplementary material of Havgaard et al. [2007] on page 87 shows that the improved energy model in version 2.1 improves the base pair sensitivity enough to be comparable with other methods.

There are other tools for making local alignments of RNAs. The most promising are Stemloc [Holmes, 2005] and CMfinder [Yao et al., 2006]. Stemloc is a SCFG based method. In principle it should perform well, but in reality it does not, (data not shown). While there has been no tests of the local alignment methods, the global tests by Gardner et al. [2005] and Dowell and Eddy [2006] also show this. CMfinder uses a Bayesian approach to make local multiple alignment. It performs well, but requires multiple sequences.

The Torarinsson et al. [2006] study shows that there are still many new RNA structures to be found, and that FOLDALIGN is one of the tools which can be used to find them. A future step will be to try to predict the functions of the newly found structures.

The ideas for future improvements to the algorithm include:

- The original FOLDALIGN algorithm was used in a multiple alignment context. The current algorithm is used to make global multiple alignment in Torarinsson et al. [2007]. It would be desirable with a method which can make local multiple alignments.

- Allowing insert base pairs at the beginning of stems. There is no biological arguments for not allowing inserts at the beginning of stems, and it would therefore be desirable to allow these. In some structural families there are stems which are not present in all structures. It would be relevant to include these insert stems.

- It is reasonable to assume that a good long alignment can have more gaps inserted than a good short alignment. A length dependent δ parameter similar to the length dependent M parameter used in Uzilov et al. [2006] could be used to improve the time and memory requirements.

- Limiting the number of stems which can be part of a multibranch point by requiring that such stems must have a score above a cutoff. If no similarity scores are used, a natural cutoff would be zero. Using this cutoff the folding energy of the stem would be zero or less which indicates a stable stem.

- In the current model an insert base pair is scored as two normal gaps. A better model would probably have one set of gap penalties for single strand gaps and one set for base pair inserts.

- The current file format is not as easy to read for a human as it should be. The future file format will probably be similar to that of the web-server.

- Better estimates of the extreme value parameters. With the current model a single shuffling of two 500 nucleotide long sequences yields ~ 240 scores for estimating the distribution. When aligning one pair of sequences, it is therefore possible to do enough shufflings to get a reasonable estimate. But for large scale searches predetermining the distribution is desirable. This could for example be done by using either a partition function [Klein and Eddy, 2003] or a support vector machine [Washietl et al., 2005b].

- Two types of heuristics are currently very popular. Anchoring using sequence similarity, and structural constraint using pre-folding of the sequences. The performance of pre-folding algorithms may be improved by also using pruning. When pruning is used, the pre-folding can allow for more low scoring base pairs while the pruning makes sure that poor structures do not slow down the algorithm too much.

Appendix A

The recursion

This chapter describes the recursion of the 2.1 version of FOLDALIGN. To make it easier to read it is written in the form usually used to describe dynamic programming recursions rather than the expansion form used in the article and the source code [Havgaard et al., 2007]. The recursion as showed here is simplified in two ways. Affine gap penalties are not included, and one base pair long stems are allowed. Otherwise the recursion is the same as the one used in version 2.1. The notation used in the recursion can be seen in Table (A.1). Figure (2.5) on page 13 shows an overview of the energy model. $D_{(i,j,k,l)}$ is the alignment score, $\sigma_{(i,j,k,l)}$ is the alignment state, $\mu_{1(i,j,k,l)}$, $\mu_{2(i,j,k,l)}$, $\mu_{3(i,j,k,l)}$, and $\mu_{4(i,j,k,l)}$ are the lengths of the single stranded regions external to the last base pair.

$$D_{(i,j,k,l)} = \max \begin{cases} D_{(i+1,j-1,k+1,l-1)} & +S_{bp}(\sigma_{(i+1,j-1,k+1,l-1)}) & \text{(a)} \\ D_{(i+1,j-1,k,l)} & +S_{bpiI}(\sigma_{(i+1,j-1,k,l)}) & \text{(b)} \\ D_{(i,j,k+1,l-1)} & +S_{bpiK}(\sigma_{(i,j,k+1,l-1)}) & \text{(c)} \\ D_{(i+1,j,k+1,l)} & +S_{al}(\sigma_{(i+1,j,k+1,l)}) & \text{(d)} \\ D_{(i,j-1,k,l-1)} & +S_{ar}(\sigma_{(i,j-1,k,l-1)}) & \text{(e)} \\ D_{(i+1,j,k,l)} & +S_{glI}(\sigma_{(i+1,j,k,l)}) & \text{(f)} \\ D_{(i,j,k+1,l)} & +S_{glK}(\sigma_{(i,j,k+1,l)}) & \text{(g)} \\ D_{(i,j-1,k,l)} & +S_{grI}(\sigma_{(i,j-1,k,l)}) & \text{(h)} \\ D_{(i,j,k,l-1)} & +S_{grK}(\sigma_{(i,j,k,l-1)}) & \text{(i)} \\ \max\limits_{\substack{i<m<j \\ k<n<l}} \{D'_{(i,m,k,n)} + D'_{(m+1,j,n+1,l)} + S_{mbl}(\sigma_l, \sigma_r)\} & & \text{(j)} \end{cases} \tag{A.1}$$

The calculation is initialized by aligning two nucleotides in the hairpin state.

$$D_{(i,i,k,k)} = R_{ss}(n_i, n_k) + L_{hp}(1,1) \tag{A.2}$$

When the score $D_{(i,j,k,l)}$ is calculated, the score of single stranded nucleotides external to the last base pair and the score of the last base pair may not be correct. It is correct by this calculation:

$$D'_{(i,j,k,l)} = D_{(i,j,k,l)} + S'(\sigma) \tag{A.3}$$

C_{mblend}	The cost of closing a multibranched loop
$C_{mblhelix}$	The cost of adding a stem to the multibranch loop (not including the first stem)
C_{mblnuc}	The cost of adding a single stranded nucleotide to a multibranched loop
C_{nGC}	The non-GC stem end cost
D	The score of an alignment. Not corrected for external single stranded nucleotides
D'	The alignment score. Corrected for external single stranded nucleotides
I sequence	One of the two sequences. Usually the longest
K sequence	The other sequence. Usually the shortest
L_{bl}	Bulge length cost
L_{il}	Internal loop length cost. Includes the asymmetry cost.
L_{hp}	Hairpin length cost
μ_1	Length of the single stranded region upstream of the last base pair in the I sequence
μ_2	Length of the single stranded region downstream of the last base pair in the I sequence
μ_3	Length of the single stranded region upstream of the last base pair in the K sequence
μ_4	Length of the single stranded region downstream of the last base pair in the K sequence
R_{bp}	Base pair substitution score
R_{ss}	Single strand substitution score
s	Stacking score
s_{hp}	Hairpin end stacking score
s_{il}	Internal loop end stacking score
σ	The state of an alignment

Table A.1: Notation

A.1 Contexts

A.1.1 Base pair

Add a base pair to both structures. The S_{bp} score is only calculated if both n_i and n_j base pair, and n_k and n_l base pair. The $\mu_{(i,j,k,l)}$ lengths are set to zero if this case has the highest score in equation (A.1).

$$S_{bp}(\sigma_{(i+1,j-1,k+1,l-1)}) = \begin{cases}
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + s_{hp}(n_{i+1}, n_{j-1}, n_i, n_j,) \\ \quad + s_{hp}(n_{k+1}, n_{l-1}, n_k, n_l) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a hairpin state.} \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\[2ex]
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + s(n_{i+1}, n_{j-1}, n_i, n_j) \\ \quad + s(n_{k+1}, n_{l-1}, n_k, n_l) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a stem or an} \\ \quad \text{insert base pair state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + s(n_{i+\mu_1+1}, n_{j-1}, n_i, n_j) \\ \quad + s(n_{k+\mu_3+1}, n_{l-1}, n_k, n_l) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a bulge left state} \\ \quad \text{and } \mu_1 \leq 1 \text{ and } \mu_3 \leq 1, \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + C_{nGC}(n_i, n_j) + C_{nGC}(n_k, n_l) \\ \quad + C_{nGC}(n_{i+\mu_1+1}, n_{j-1}) \\ \quad + C_{nGC}(n_{k+\mu_3+1}, n_{l-1}) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a bulge left state} \\ \quad \text{and } \mu_1 > 1 \text{ or } \mu_3 > 1, \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\[2ex]
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + s(n_{i+1}, n_{j-\mu_2-1}, n_i, n_j) \\ \quad + s(n_{k+1}, n_{l-\mu_4-1}, n_k, n_l) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a bulge right state} \\ \quad \text{and } \mu_2 \leq 1 \text{ and } \mu_4 \leq 1, \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + C_{nGC}(n_i, n_j) + C_{nGC}(n_k, n_l) \\ \quad + C_{nGC}(n_{i+1}, n_{j-\mu_2-1}) \\ \quad + C_{nGC}(n_{k+1}, n_{l-\mu_4-1}) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a bulge right state} \\ \quad \text{and } \mu_2 > 1 \text{ or } \mu_4 > 1, \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\[2ex]
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) \\ \quad + s_{il}(n_{i+1}, n_{j-1}, n_i, n_j) \\ \quad + s_{il}(n_{i+\mu_1}, n_{j-\mu_2}, n_{i+\mu_1+1}, n_{j-\mu_2-1}) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is an internal loop} \\ \quad \text{state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array} \\
\begin{array}{l} R_{bp}(n_i, n_j, n_k, n_l) + C_{mblend} \\ \quad + C_{nGC}(n_i, n_j) + C_{nGC}(n_k, n_l) \end{array} & \begin{array}{l} \text{if } \sigma \text{ is a left or right} \\ \quad \text{multibranch loop state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a stem} \\ \quad \text{state.} \end{array}
\end{cases}$$

A.1.2 Insert base pair

Insert a base pair in a stem from the I sequence. The S_{bpiI} score is only calculated when n_i and n_j base pair, and the state is stem or insert base pair in the I sequence. The $\mu_{(i,j,k,l)}$ lengths are all set to zero if this case has the highest score in equation (A.1).

$$S_{bpiI}(\sigma_{(i+1,j-1,k,l)}) = 2 \times R_{ss}(gap) + s(n_{i+1}, n_{j-1}, n_i, n_j) \quad \sigma_{(i,j,k,l)} \text{ becomes an insert base pair in the } I \text{ sequence state.}$$

Insert a base pair in a stem from the K sequence. The S_{bpiK} score is only calculated when n_k and n_l base pair, and the state is stem or insert base pair in the K sequence. The $\mu_{(i,j,k,l)}$ lengths are all set to zero if this case has the highest score in equation (A.1).

$$S_{bpiK}(\sigma_{(i,j,k+1,l-1)}) = 2 \times R_{ss}(gap) + s(n_{k+1}, n_{l-1}, n_k, n_l) \quad \sigma_{(i,j,k,l)} \text{ becomes an insert base pair in the } K \text{ sequence state.}$$

A.1.3 Align left

Align two single stranded nucleotides on the left side of an alignment.

$$S_{al}(\sigma_{(i+1,j,k+1,l)}) = \begin{cases} R_{ss}(n_i,n_k) - L_{hp}(\mu_1) - L_{hp}(\mu_3) + L_{hp}(\mu_1+1) + L_{hp}(\mu_3+1) & \text{if } \sigma \text{ is a hairpin state, } \sigma_{(i,j,k,l)} \text{ becomes a hairpin state.} \\ R_{ss}(n_i,n_k) + 2 \times L_{bl}(1) & \text{if } \sigma \text{ is a stem state or an insert base pair state, } \sigma_{(i,j,k,l)} \text{ becomes a bulge left state.} \\ R_{ss}(n_i,n_k) - L_{bl}(\mu_1) - L_{bl}(\mu_3) + L_{bl}(\mu_1+1) + L_{bl}(\mu_3+1) & \text{if } \sigma \text{ is a bulge left state, } \sigma_{(i,j,k,l)} \text{ becomes a bulge left state.} \\ R_{ss}(n_i,n_k) - L_{bl}(\mu_2) - L_{bl}(\mu_4) + L_{il}(1,\mu_2) + L_{il}(1,\mu_4) & \text{if } \sigma \text{ is a bulge right state, } \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ R_{ss}(n_i,n_k) - L_{il}(\mu_1,\mu_2) - L_{il}(\mu_3,\mu_4) + L_{il}(\mu_1+1,\mu_2) + L_{il}(\mu_3+1,\mu_4) & \text{if } \sigma \text{ is an internal loop state, } \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ R_{ss}(n_i,n_k) + 2 \times C_{mblnuc} & \text{if } \sigma \text{ is a left or right multibranch loop state, } \sigma_{(i,j,k,l)} \text{ becomes a left multibranch loop state.} \end{cases}$$

If this case wins in equation (A.1), the lengths are updated like this:

$$\begin{aligned}
\mu_{1(i,j,k,l)} &= \mu_{1(i+1,j,k+1,l)} + 1\\
\mu_{2(i,j,k,l)} &= \mu_{2(i+1,j,k+1,l)}\\
\mu_{3(i,j,k,l)} &= \mu_{3(i+1,j,k+1,l)} + 1\\
\mu_{4(i,j,k,l)} &= \mu_{4(i+1,j,k+1,l)}
\end{aligned}$$

A.1.4 Align right

Align two single stranded nucleotides on the right side of an alignment.

$$S_{ar}(\sigma_{(i,j-1,k,l-1)}) = \begin{cases}
\begin{array}{l} R_{ss}(n_j, n_l) \\ \quad - L_{hp}(\mu_1) - L_{hp}(\mu_3) \\ \quad + L_{hp}(\mu_1+1) + L_{hp}(\mu_3+1) \end{array} & \begin{array}{l}\text{if } \sigma \text{ is a hairpin state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a} \\ \quad \text{hairpin state.}\end{array} \\[2ex]
\begin{array}{l} R_{ss}(n_j, n_l) \\ \quad + 2 \times L_{bl}(1) \end{array} & \begin{array}{l}\text{if } \sigma \text{ is a stem state} \\ \quad \text{or an insert base pair state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a bulge right state.}\end{array} \\
\begin{array}{l} R_{ss}(n_j, n_l) \\ \quad - L_{bl}(\mu_1) - L_{bl}(\mu_3) \\ \quad + L_{il}(\mu_1, 1) + L_{il}(\mu_3, 1) \end{array} & \begin{array}{l}\text{if } \sigma \text{ is a bulge left state,} \\ \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.}\end{array} \\[2ex]
\begin{array}{l} R_{ss}(n_j, n_l) \\ \quad - L_{bl}(\mu_2) - L_{bl}(\mu_4) \\ \quad + L_{bl}(\mu_2+1) + L_{bl}(\mu_4+1) \end{array} & \begin{array}{l}\text{if } \sigma \text{ is a bulge right state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a bulge right state.}\end{array} \\[2ex]
\begin{array}{l} R_{ss}(n_j, n_l) \\ \quad - L_{il}(\mu_1, \mu_2) - L_{il}(\mu_3, \mu_4) \\ \quad + L_{il}(\mu_1, \mu_2+1) \\ \quad + L_{il}(\mu_3, \mu_4+1) \end{array} & \begin{array}{l}\text{if } \sigma \text{ is an internal loop state,} \\ \sigma_{(i,j,k,l)} \text{ becomes an} \\ \quad \text{internal loop state.}\end{array} \\[2ex]
R_{ss}(n_j, n_l) + 2 \times C_{mblnuc} & \begin{array}{l}\text{if } \sigma \text{ is a left multibranch loop state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a left} \\ \quad \text{multibranch loop state.}\end{array} \\
R_{ss}(n_j, n_l) + 2 \times C_{mblnuc} & \begin{array}{l}\text{if } \sigma \text{ is a right multibranch loop state,} \\ \sigma_{(i,j,k,l)} \text{ becomes a right} \\ \quad \text{multibranch loop state.}\end{array}
\end{cases}$$

If this case wins in equation (A.1), the lengths are updated like this:

$$\begin{aligned}
\mu_{1(i,j,k,l)} &= \mu_{1(i,j-1,k,l-1)}\\
\mu_{2(i,j,k,l)} &= \mu_{2(i,j-1,k,l-1)} + 1\\
\mu_{3(i,j,k,l)} &= \mu_{3(i,j-1,k,l-1)}\\
\mu_{4(i,j,k,l)} &= \mu_{4(i,j-1,k,l-1)} + 1
\end{aligned}$$

A.1.5 Gap left I

Extend an alignment with one single stranded nucleotide on the left side of the I sequence.

$$S_{glI}(\sigma_{(i+1,j,k,l)}) = \begin{cases} R_{ss}(gap) & \text{if } \sigma \text{ is a hairpin state,} \\ \quad - L_{hp}(\mu_1) + L_{hp}(\mu_1+1) & \sigma_{(i,j,k,l)} \text{ becomes a hairpin state.} \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is a stem state} \\ \quad + L_{bl}(1) + L_{bl}(0) & \quad \text{or an insert base pair state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a bulge left state.} \\ R_{ss}(gap) & \text{if } \sigma \text{ is a bulge left state,} \\ \quad - L_{bl}(\mu_1) - L_{bl}(\mu_1+1) & \sigma_{(i,j,k,l)} \text{ becomes a bulge left state.} \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is a bulge right state,} \\ \quad - L_{bl}(\mu_2) - L_{bl}(\mu_4) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ \quad + L_{il}(1,\mu_2) + L_{il}(0,\mu_4) & \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is an internal loop state,} \\ \quad - L_{il}(\mu_1,\mu_2) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ \quad + L_{il}(\mu_1+1,\mu_2) & \\ & \\ R_{ss}(gap) + C_{mblnuc} & \text{if } \sigma \text{ is a left or right multibranch} \\ & \quad \text{loop state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a left} \\ & \quad \text{multibranch loop state.} \end{cases}$$

If this case wins in equation (A.1), the lengths are updated like this:

$$\begin{aligned} \mu_{1(i,j,k,l)} &= \mu_{1(i+1,j,k,l)} + 1 \\ \mu_{2(i,j,k,l)} &= \mu_{2(i+1,j,k,l)} \\ \mu_{3(i,j,k,l)} &= \mu_{3(i+1,j,k,l)} \\ \mu_{4(i,j,k,l)} &= \mu_{4(i+1,j,k,l)} \end{aligned}$$

A.1.6 Gap left K

Extend an alignment with one single stranded nucleotide on the left side of the K sequence.

$$S_{glK}(\sigma_{(i,j,k+1,l)}) = \begin{cases} R_{ss}(gap) & \text{if } \sigma \text{ is a hairpin state,} \\ \quad - L_{hp}(\mu_3) + L_{hp}(\mu_3+1) & \sigma_{(i,j,k,l)} \text{ becomes a hairpin state.} \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is a stem state} \\ \quad + L_{bl}(0) + L_{bl}(1) & \quad \text{or an insert base pair state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a bulge left state.} \\ R_{ss}(gap) & \text{if } \sigma \text{ is a bulge left state,} \\ \quad - L_{bl}(\mu_3) + L_{bl}(\mu_3+1) & \sigma_{(i,j,k,l)} \text{ becomes a bulge left state.} \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is a bulge right state,} \\ \quad - L_{bl}(\mu_2) - L_{bl}(\mu_4) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ \quad + L_{il}(0,\mu_2) + L_{il}(1,\mu_4) & \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is an internal loop state,} \\ \quad - L_{il}(\mu_3,\mu_4) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ \quad + L_{il}(\mu_3+1,\mu_4) & \\ & \\ R_{ss}(gap) + C_{mblnuc} & \text{if } \sigma \text{ is a left or right multibranch} \\ & \quad \text{loop state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a left multibranch} \\ & \quad \text{loop state.} \end{cases}$$

If this case wins in equation (A.1), the lengths are updated like this:

$$\begin{aligned} \mu_{1(i,j,k,l)} &= \mu_{1(i,j,k+1,l)} \\ \mu_{2(i,j,k,l)} &= \mu_{2(i,j,k+1,l)} \\ \mu_{3(i,j,k,l)} &= \mu_{3(i,j,k+1,l)} + 1 \\ \mu_{4(i,j,k,l)} &= \mu_{4(i,j,k+1,l)} \end{aligned}$$

A.1.7 Gap right I

Extend an alignment with one single stranded nucleotide on the right side of the I sequence.

$$
S_{grI}(\sigma_{(i,j-1,k,l)}) = \begin{cases}
R_{ss}(gap) & \text{if } \sigma \text{ is a hairpin state,} \\
\quad - L_{hp}(\mu_1) + L_{hp}(\mu_1 + 1) & \sigma_{(i,j,k,l)} \text{ becomes a hairpin state.} \\
 & \\
R_{ss}(gap) & \text{if } \sigma \text{ is a stem state} \\
\quad + L_{bl}(1) + L_{bl}(0) & \quad \text{or an insert base pair state,} \\
 & \sigma_{(i,j,k,l)} \text{ becomes a bulge right state.} \\
R_{ss}(gap) & \text{if } \sigma \text{ is a bulge left state,} \\
\quad - L_{bl}(\mu_1) - L_{bl}(\mu_3) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\
\quad + L_{il}(\mu_1, 1) + L_{il}(\mu_3, 0) & \\
 & \\
R_{ss}(gap) & \text{if } \sigma \text{ is a bulge right state,} \\
\quad - L_{bl}(\mu_2) + L_{bl}(\mu_2 + 1) & \sigma_{(i,j,k,l)} \text{ becomes a bulge right state.} \\
 & \\
R_{ss}(gap) & \text{if } \sigma \text{ is an internal loop state,} \\
\quad - L_{il}(\mu_1, \mu_2) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\
\quad + L_{il}(\mu_1, \mu_2 + 1) & \\
 & \\
R_{ss}(gap) + C_{mblnuc} & \text{if } \sigma \text{ is a left multibranch loop state,} \\
 & \sigma_{(i,j,k,l)} \text{ becomes a left multibranch} \\
 & \quad \text{loop state.} \\
R_{ss}(gap) + C_{mblnuc} & \text{if } \sigma \text{ is a right multibranch loop state,} \\
 & \sigma_{(i,j,k,l)} \text{ becomes a right multibranch} \\
 & \quad \text{loop state.}
\end{cases}
$$

If this case wins in equation (A.1), the lengths are updated like this:

$$
\begin{aligned}
\mu_{1(i,j,k,l)} &= \mu_{1(i,j-1,k,l)} \\
\mu_{2(i,j,k,l)} &= \mu_{2(i,j-1,k,l)} + 1 \\
\mu_{3(i,j,k,l)} &= \mu_{3(i,j-1,k,l)} \\
\mu_{4(i,j,k,l)} &= \mu_{4(i,j-1,k,l)}
\end{aligned}
$$

A.1.8 Gap right K

Extend an alignment with one single stranded nucleotide on the right side of the K sequence.

$$S_{grK}(\sigma_{(i,j,k,l-1)}) = \begin{cases} R_{ss}(gap) & \text{if } \sigma \text{ is a hairpin state,} \\ \quad - L_{hp}(\mu_3) + L_{hp}(\mu_3+1) & \sigma_{(i,j,k,l)} \text{ becomes a hairpin state.} \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is a stem state} \\ \quad + L_{bl}(0) + L_{bl}(1) & \quad \text{or an insert base pair state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a bulge right state.} \\ R_{ss}(gap) - L_{bl}(\mu_1,\mu_3) & \text{if } \sigma \text{ is a bulge left state,} \\ \quad - L_{bl}(\mu_1) - L_{bl}(\mu_3) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ \quad + L_{il}(\mu_1,0) + L_{il}(\mu_3,1) & \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is a bulge right state,} \\ \quad - L_{bl}(\mu_4) + L_{bl}(\mu_4+1) & \sigma_{(i,j,k,l)} \text{ becomes a bulge right state.} \\ & \\ R_{ss}(gap) & \text{if } \sigma \text{ is an internal loop state,} \\ \quad - L_{il}(\mu_3,\mu_4) & \sigma_{(i,j,k,l)} \text{ becomes an internal loop state.} \\ \quad + L_{il}(\mu_3,\mu_4+1) & \\ & \\ R_{ss}(gap) + C_{mblnuc} & \text{if } \sigma \text{ is a left multibranch loop state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a left multibranch} \\ & \quad \text{loop state.} \\ R_{ss}(gap) + C_{mblnuc} & \text{if } \sigma \text{ is a right multibranch loop state,} \\ & \sigma_{(i,j,k,l)} \text{ becomes a right multibranch} \\ & \quad \text{loop state.} \end{cases}$$

If this case wins in equation (A.1), the lengths are updated like this:

$$\begin{aligned} \mu_{1(i,j,k,l)} &= \mu_{1(i,j,k,l-1)} \\ \mu_{2(i,j,k,l)} &= \mu_{2(i,j,k,l-1)} \\ \mu_{3(i,j,k,l)} &= \mu_{3(i,j,k,l-1)} \\ \mu_{4(i,j,k,l)} &= \mu_{4(i,j,k,l-1)} + 1 \end{aligned}$$

A.1.9 Multibranched loops

Join two alignments to get a multibranched structure. In equation A.1 case (j) is only calculated when σ_l is a stem, base pair insert, bulge right, or a right multibranch loop state, and σ_r is a stem or a base pair insert state, see Chapter 2.3 and Figure 2.11. The $\mu_{i,j,k,l}$ lengths are set to zero if this case has the highest alignment score. Remember that the $D_{ij,kl}$ is not used directly in this calculation. It is always the external loop version $D'_{ij,kl}$ which is used, see the External nucleotides section below.

$$S_{mbl}(\sigma_l, \sigma_r) = C_{mblhelix} \quad \sigma_{(i,j,k,l)} \text{ always becomes a right multibranch loop state.}$$

A.1.10 External nucleotides

Single stranded nucleotides external to all base pairs must be scored like single stranded nucleotides in multibranched loops. The score must therefore be recalculated when the alignment state is one of the hairpin, bulge, or internal loop states. Furthermore the cost for non-GC base pairs must also be added in cases where the alignment state is one of the base pair states. This calculation does not affect the state or the μ lengths of the alignment.

$$S'(\sigma_{(i,j,k,l)}) = \begin{cases} (\mu_1+\mu_3)\times C_{mblnuc} - L_{hp}(\mu_1) - L_{hp}(\mu_3) & \text{if } \sigma \text{ is a hairpin state.} \\ C_{nGC}(n_i,n_j) + C_{nGC}(n_k,n_l) & \text{if } \sigma \text{ is a stem or a base pair insert state.} \\ C_{nGC}(n_{i+\mu_1},n_j) + C_{nGC}(n_{k+\mu_3},n_l) \\ \quad + (\mu_1+\mu_3)\times C_{mblnuc} - L_{bl}(\mu_1) - L_{bl}(\mu_3) & \text{if } \sigma \text{ is a bulge left state.} \\ C_{nGC}(n_i,n_{j-\mu_2}) + C_{nGC}(n_k,n_{l-\mu_4}) \\ \quad + (\mu_2+\mu_4)\times C_{mblnuc} - L_{bl}(\mu_2) - L_{bl}(\mu_4) & \text{if } \sigma \text{ is a bulge right state.} \\ C_{nGC}(n_{i+\mu_1},n_{j-\mu_2}) + C_{nGC}(n_{k+\mu_3},n_{l-\mu_4}) \\ \quad + (\mu_1+\mu_2+\mu_3+\mu_4)\times C_{mblnuc} \\ \quad - L_{il}(\mu_1,\mu_2) - L_{il}(\mu_3,\mu_4) & \text{if } \sigma \text{ is an internal loop state.} \\ 0 & \text{if } \sigma \text{ is a right or left multibranch state} \end{cases}$$

Bibliography

Altschul, S. (1991). Amino acid substitution matrices from an information theoretic perspective. *J Mol Biol.*, 219(3):555–65.

Altschul, S., Bundschuh, R., Olsen, R., and Hwa, T. (2001). The estimation of statistical parameters for local alignment score distributions. *Nucleic Acids Res.*, 29(2):351–61.

Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *J Mol Biol.*, 215(3):403–10.

Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25(17):3389–402.

Benson, D., Karsch Mizrachi, I., Lipman, D., Ostell, J., and Wheeler, D. (2004). GenBank: update. *Nucleic Acids Res.*, 32:23–6.

Brown, J. (1999). The Ribonuclease P Database. *Nucleic Acids Res.*, 27(1):314.

Chvátal, V. and Sankoff, D. (1975). Longest common subsequences of two random sequences. *Journal of Applied Probabillity.*, 12:306–315.

DeVoe, H. and Tinoco Jr., I. (1962). The stability of helical polynucleotides: base contributions. *J Mol Biol.*, 4:500–17.

Dowell, R. and Eddy, S. (2006). Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics.*, 7:400.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis*. Cambridge University Press, Cambridge, United Kingdom.

Eddy, S. (2002). A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics.*, 3(1):18.

Eddy, S. and Durbin, R. (1994). RNA sequence analysis using covariance models. *Nucleic Acids Res*, 22(11):2079–88.

Gardner, P. and Giegerich, R. (2004). A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics.*, 5:140.

Gardner, P., Wilm, A., and Washietl, S. (2005). A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res.*, 33(8):2433–9.

Gorodkin, J., Havgaard, J., Enstero, M., Sawera, M., Jensen, P., Öhman, M., and Fredholm, M. (2006). MicroRNA sequence motifs reveal asymmetry between the stem arms. *Computational Biology and Chemistry*, 30(4):249–54.

Gorodkin, J., Heyer, L., Brunak, S., and Stormo, G. (1997a). Displaying the information contents of structural RNA alignments: the structure logos. *Comput Appl Biosci.*, 13(6):583–6.

Gorodkin, J., Heyer, L., and Stormo, G. (1997b). Finding common sequence and structure motifs in a set of RNA sequences. *Proc Int Conf Intell Syst Mol Biol*, 5:120–3.

Gorodkin, J., Heyer, L., and Stormo, G. (1997c). Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res*, 25(18):3724–32.

Gorodkin, J., Lyngsø, R., and Stormo, G. (2001a). A mini-greedy algorithm for faster structural RNA stem-loop search. *Genome Inform Ser Workshop Genome Inform.*, 12:184–93.

Gorodkin, J., Stricklin, S., and Stormo, G. (2001b). Discovering common stem-loop motifs in unaligned RNA sequences. *Nucleic Acids Res*, 29(10):2135–44.

Griffiths Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, S. (2003). Rfam: an RNA family database. *Nucleic Acids Res*, 31(1):439–41.

Griffiths Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S., and Bateman, A. (2005). Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.*, 33(1):D121–4.

Gumbel, E. J. (1958). *Statistics of Extremes*. Columbia University Press.

Harris, J., Haas, E., Williams, D., Frank, D., and Brown, J. (2001). New insight into RNase P RNA structure from comparative analysis of the archaeal RNA. *RNA.*, 7(2):220–32.

Havgaard, J., Lyngsø, R., and Gorodkin, J. (2005a). The FOLDALIGN web server for pairwise structural RNA alignment and mutual motif search. *Nucleic Acids Res.*, 33:W650–3.

Havgaard, J., Lyngsø, R., Stormo, G., and Gorodkin, J. (2005b). Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics.*, 21(9):1815–24.

Havgaard, J., Torarinsson, E., and Gorodkin, J. (2007). Fast pairwise local structural RNA alignments by pruning of the dynamical programming matrix. *Submitted.*

Henikoff, S. and Henikoff, J. (1992). Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A.*, 89(22):10915–9.

Heyer, L. J. (2000). A generalized Erdös-rényi law for sequence analysis problems. *Methodology and Computing in Applied Probability*, 2:309–329.

Hirschberg, D. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18:341–343.

Hofacker, I. (2003). Vienna RNA secondary structure server. *Nucleic Acids Res.*, 31(13):3429–31.

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

Hofacker, I., Fekete, M., and Stadler, P. (2002). Secondary structure prediction for aligned RNA sequences. *J Mol Biol.*, 319(5):1059–66.

Hofacker, I., Fontana, W., Stadler, P., Bonhoeffer, S., Tacker, M., and Schuster, P. (1994). Fast Folding and Comparison of RNA Secondary Structures. *Monatshefte f. Chemie.*, 125:167–188.

Holmes, I. (2005). Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics.*, 6:73.

Huttenhofer, A., Brosius, J., and Bachellerie, J. (2002). RNomics: identification and function of small, non-messenger RNAs. *Curr Opin Chem Biol.*, 6(6):835–43.

Huttenhofer, A., Schattner, P., and Polacek, N. (2005). Non-coding RNAs: hope or hype? *Trends Genet.*, 21(5):289–97.

Huttenhofer, A. and Vogel, J. (2006). Experimental approaches to identify non-coding RNAs. *Nucleic Acids Res.*, 34(2):635–46.

Karlin, S. and Altschul, S. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci U S A.*, 87(6):2264–8.

Klein, R. and Eddy, S. (2003). RSEARCH: Finding homologs of single structured RNA sequences. *BMC Bioinformatics.*, 4(1):44.

Knudsen, B. and Hein, J. (2003). Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res.*, 31(13):3423–8.

Lagos Quintana, M., Rauhut, R., Lendeckel, W., and Tuschl, T. (2001). Identification of novel genes coding for small expressed RNAs. *Science.*, 294(5543):853–8.

Lau, N., Lim, L., Weinstein, E., and Bartel, D. (2001). An abundant class of tiny RNAs with probable regulatory roles in Caenorhabditis elegans. *Science.*, 294(5543):858–62.

Lee, R. and Ambros, V. (2001). An extensive class of small RNAs in Caenorhabditis elegans. *Science*, 294(5543):862–4.

Lee, R., Feinbaum, R., and Ambros, V. (1993). The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell.*, 75(5):843–54.

Mathews, D., Disney, M., Childs, J., Schroeder, S., Zuker, M., and Turner, D. (2004). Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc Natl Acad Sci U S A.*, 101(19):7287–92.

Mathews, D., Sabina, J., Zuker, M., and Turner, D. (1999). Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol*, 288(5):911–40.

Mathews, D. and Turner, D. (2002). Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J Mol Biol.*, 317(2):191–203.

Mattick, J. (2004). RNA regulation: a new genetics? *Nat Rev Genet.*, 5(4):316–23.

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

Myers, E. and Miller, W. (1988). Optimal alignments in linear space. *Comput Appl Biosci.*, 4(1):11–7.

Nussinov, R. and Jacobson, A. (1980). Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proc Natl Acad Sci U S A*, 77(11):6903–13.

Nussinov, R., Pieczenik, G., Griggs, J. R., and Kleitman, D. J. (1978). Algorithms for loop matchings. *SIAM J. Appl. Math*, 35(1):68–82.

Olsen, R., Bundschuh, R., and Hwa, T. (1999). Rapid assessment of extremal statistics for gapped local alignment. *Proc Int Conf Intell Syst Mol Biol.*, pages 211–22.

Pearson, W. and Lipman, D. (1988). Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A.*, 85(8):2444–8.

Pedersen, J. S., Bejerano, G., Siepel, A., Rosenbloom, K., Lindblad-Toh, K., Lander, E. S., Kent, J., Miller, W., and Haussler, D. (2006). Identification and classification of conserved rna secondary structures in the human genome. *PLOS Computational Biology*, 2(4):e33.

Reeder, J. and Giegerich, R. (2005). Consensus shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction. *Bioinformatics.*, 21(17):3516–23.

Rivas, E. and Eddy, S. (2001). Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(1):8.

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

Sankoff, D. (2000). The early introduction of dynamic programming into computational biology. *Bioinformatics.*, 16(1):41–7.

Schneider, T. and Stephens, R. (1990). Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res.*, 18(20):6097–100.

Sprinzl, M., Horn, C., Brown, M., Ioudovitch, A., and Steinberg, S. (1998). Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Res.*, 26(1):148–53.

Sprinzl, M. and Vassilenko, K. (2005). Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Res.*, 33(Database issue):D139–40.

Suzuki, M. and Hayashizaki, Y. (2004). Mouse-centric comparative transcriptomics of protein coding and non-coding RNAs. *Bioessays.*, 26(8):833–43.

Szymanski, M., Barciszewska, M., Erdmann, V., and Barciszewski, J. (2002). 5S Ribosomal RNA Database. *Nucleic Acids Res.*, 30(1):176–8.

Taneda, A. (2005). Cofolga: a genetic algorithm for finding the common folding of two RNAs. *Comput Biol Chem.*, 29(2):111–9.

Tinoco Jr., I., Borer, P., Dengler, B., Levin, M., Uhlenbeck, O., Crothers, D., and Bralla, J. (1973). Improved estimation of secondary structure in ribonucleic acids. *Nat New Biol.*, 246(150):40–1.

Tinoco Jr., I., Uhlenbeck, O. C., and Levine, M. D. (1971). Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367.

Torarinsson, E., Havgaard, J., and Gorodkin, J. (2007). Multiple structural alignment and clustering of RNA sequences. *Accepted for publication in Bioinformatics*.

Torarinsson, E., Sawera, M., Havgaard, J., Fredholm, M., and Gorodkin, J. (2006). Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure. *Genome Res.*, 16(7):885–9.

Uzilov, A., Keegan, J., and Mathews, D. (2006). Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics.*, 7(1):173.

Van de Peer, Y., Van den Broeck, I., De Rijk, P., and De Wachter, R. (1994). Database on the structure of small ribosomal subunit RNA. *Nucleic Acids Res.*, 22(17):3488–94.

Walczak, R., Westhof, E., Carbon, P., and Krol, A. (1996). A novel RNA structural motif in the selenocysteine insertion element of eukaryotic selenoprotein mRNAs. *RNA.*, 2(4):367–79.

Wang, T. and Stormo, G. (2003). Combining phylogenetic data with co-regulated genes to identify regulatory motifs. *Bioinformatics.*, 19(18):2369–80.

Washietl, S. and Hofacker, I. (2004). Consensus folding of aligned sequences as a new measure for the detection of functional RNAs by comparative genomics. *J Mol Biol.*, 342(1):19–30.

Washietl, S., Hofacker, I., Lukasser, M., Huttenhofer, A., and Stadler, P. (2005a). Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. *Nat Biotechnol.*, 23(11):1383–90.

Washietl, S., Hofacker, I., and Stadler, P. (2005b). Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci U S A.*, 102(7):2454–9.

Waterman, M. S. (1978). Secondary structure of single stranded nucleic acids. *Adv. Math. Suppl. Stud.*, 1:167–212.

Waterman, M. S. and Smith, T. F. (1978). Secondary structure of single stranded nucleic acids. *Math. Biosci.*, 42:257–266.

Waterman, M. S. and Vingron, M. (1994). Sequence Comparison Significance and Poisson Approximation. *Statistical Science*, 9(3):367–381.

Winkler, W. (2005). Riboswitches and the role of noncoding RNAs in bacterial metabolic control. *Curr Opin Chem Biol.*, 9(6):594–602.

Woodson, S. (2005). Structure and assembly of group I introns. *Curr Opin Struct Biol.*, 15(3):324–30.

Xia, T., SantaLucia, J. J., Burkard, M., Kierzek, R., Schroeder, S., Jiao, X., Cox, C., and Turner, D. (1998). Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry.*, 37(42):14719–35.

Yao, Z., Weinberg, Z., and Ruzzo, W. (2006). CMfinder–a covariance model based RNA motif finding algorithm. *Bioinformatics.*, 22(4):445–52.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, 31(13):3406–15.

Zwieb, C. (1996). The uRNA database. *Nucleic Acids Res.*, 24(1):76–9.

Article 1-5 and letter 1 in this PhD has been removed, due to restrictions from the publishers of the journals in which these works were published. The articles and the letter can be found here:

Article 1:
Havgaard, J. H., Lyngso, R. B., Stormo, G. D., & Gorodkin, J. (2005). Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics,* 21, 1815-1824.
DOI: 10.1093/bioinformatics/bti279

Article 2:
Havgaard, J. H., Lyngso, R. B., & Gorodkin, J. (2005). The FOLDALIGN web server for pairwise structural RNA alignment and mutual motif search. *Nucleic Acids Research*, 33, W650-W653.
DOI: 10.1093/nar/gki473

Article 3:
Havgaard, J. H., Torarinsson, E., & Gorodkin, J. (2007). Fast Pairwise Structural RNA Alignments by Pruning of the Dynamical Programming Matrix. *PLoS Comput Biol*, 3, e193.
DOI: 10.1371/journal.pcbi.0030193

Article 4:
Torarinsson, E., Havgaard, J. H., & Gorodkin, J. (2007). Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, 23, 926-932.
DOI: 10.1093/bioinformatics/btm049

Article 5:
Gorodkin, J., Havgaard, J. H., Ensterö, M., Sawera, M., Jensen, P., Öhman, M. et al. (2006). MicroRNA sequence motifs reveal asymmetry between the stem arms. *Computational Biology and Chemistry*, 30, 249-254.
DOI: 10.1016/j.compbiolchem.2006.04.006

Letter 1:
Torarinsson, E., Sawera, M., Havgaard, J. H., Fredholm, M., & Gorodkin, J. (2006). Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure. *Genome Research*, 16, 885-889.
DOI: 10.1101/gr.5226606

Havgaard et al. [2005b]: Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%

Havgaard et al. [2005a]: The FOLDALIGN web server for pairwise structural RNA alignment and mutual motif search

Havgaard et al. [2005a]: Supplementary material

References

[1] J.H. Havgaard, R. Lyngsø, G.D. Stormo, and J. Gorodkin. Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics*, 10.1093/bioinformatics/bti279, 2005.

[2] B.W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta*, 405(2):442–451, 1975.

[3] M. Szymanski, M.Z. Barciszewska, V.A. Erdmann, and J. Barciszewski. 5S Ribosomal RNA Database. *Nucleic Acids Research*, 30(1):176–8, 2002.

[4] M. Sprinzl, C. Horn, M. Brown, A. Ioudovitch, and S. Steinberg. Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Research*, 26(1):148–53, 1998.

[5] C. Zwieb. The uRNA database. *Nucleic Acids Research*, 24(1):76–9, 1996.

[6] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of representative protein data sets. *Protein Science*, 1(3):409–17, 1992.

[7] D.H. Mathews and D.H. Turner. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology*, 317(2):191–203, 2002.

[8] M.A. Rosenblad, J. Gorodkin, B. Knudsen, C. Zwieb, and T. Samuelsson. SRPDB: Signal Recognition Particle Database. *Nucleic Acids Research*, 31(1):363–4, 2003.

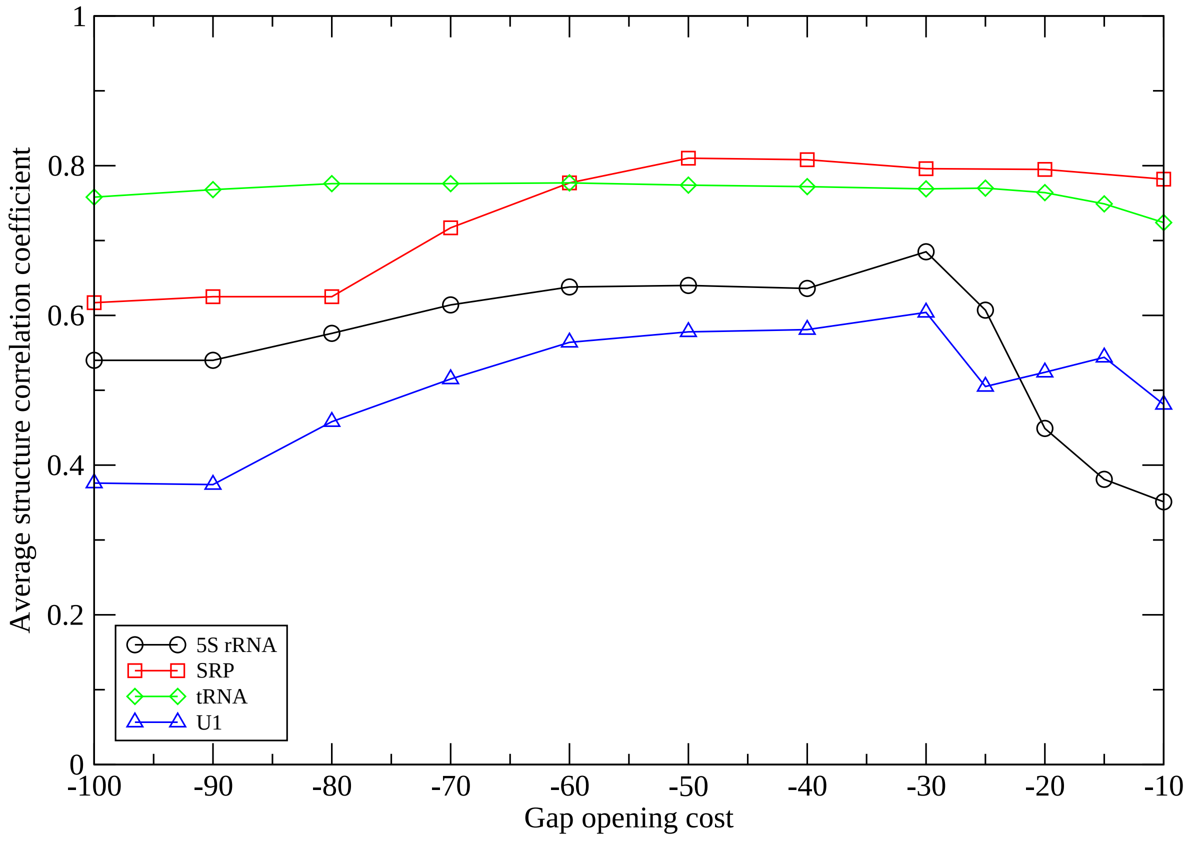


Figure S1: The average performance as function of the gap opening cost. In addition to the performance for low similarity 5S rRNA, tRNA, and U1 data the performance for the stem-loop SRP data is also shown.

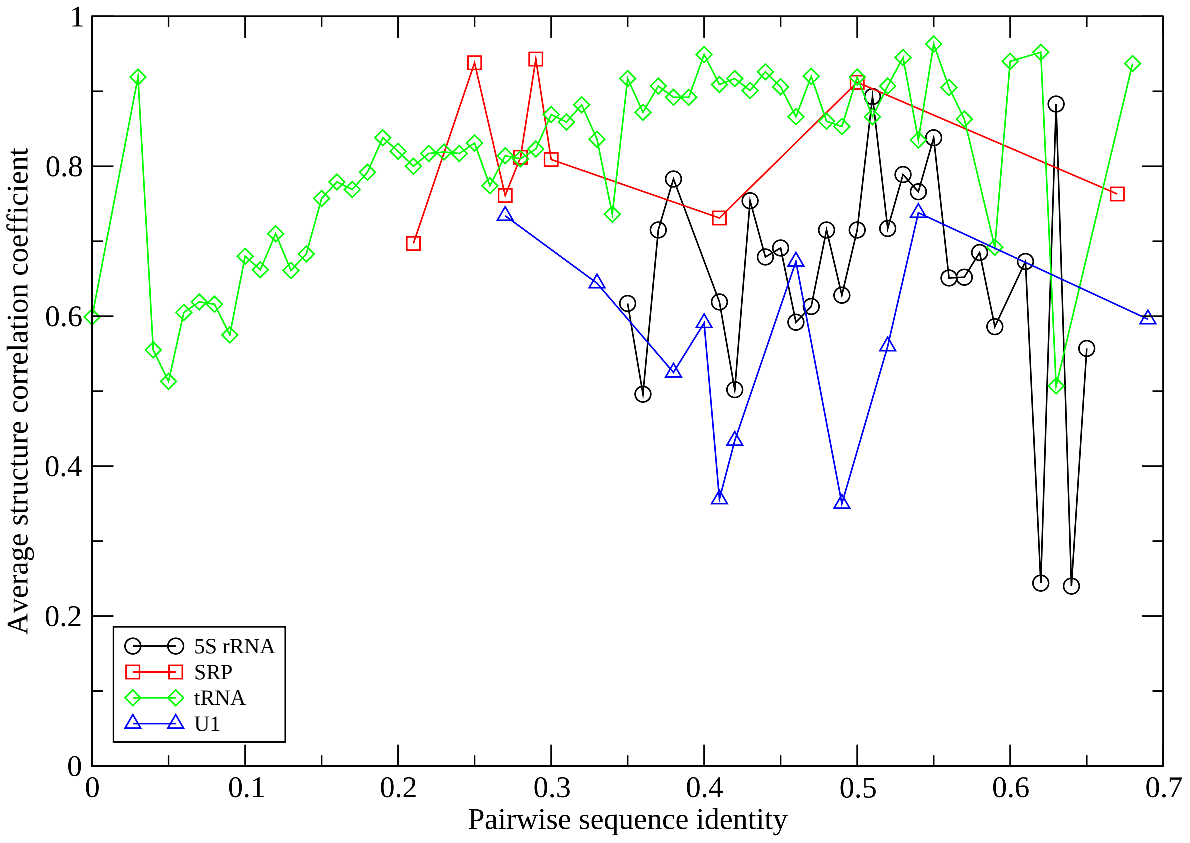


Figure S2: Performance as a function of sequence identity. The optimal gap penalties for 5S rRNA and U1 were: Gap opening -30 and gap elongation -15. For tRNA they were: Gap opening -60 and gap elongation -30. For the stem-loop SRPs the gap opening cost was -50, and the gap elongation cost was -25. The fluctuations in performance for 5S rRNA, SRP, and U1 are likely to be due to the limited amount of data at these identities. This is also true for the low and high similarity parts of the tRNA curve.

Havgaard et al. [2007]: Fast pairwise local structural RNA alignments by pruning of the dynamical programming matrix

Havgaard et al. [2007]: Supplementary material

Supplementary material for Fast pairwise local structural RNA alignments by pruning of the dynamical programming matrix

Jakob H. Havgaard[1], Elfar Torarinsson[1,2], Jan Gorodkin[1]
[1] Division of Genetics and Bioinformatic, IBHV, University of Copenhagen, Grønnegårdsvej 3, 1870 Frederiksberg C, Denmark
[2] Department of Natural Sciences, University of Copenhagen, Thorvaldsensvej 40, 1870 Frederiksberg C, Denmark

Global alignment

Initial tests of FOLDALIGN's global alignment performance while using pruning showed that the simple pruning used during local alignment cuts away to much during global alignment. The problem is that the global alignment must insert a minimum number of gaps equal to the length difference between the two sequences. When the length difference is large, the cost of inserting the minimum number of gaps is enough to make the algorithm prune away all alignments. To circumvent this problem a global alignment pruning scheme is used:

$$\begin{aligned} D_{ij,kl} &< \Theta_{global} \\ \Theta_{global} &= \Theta_{local} - G_E \times \min\{\text{abs}(l_I - l_K), \\ &\qquad \text{abs}(L_I - L_K)\} \end{aligned} \quad (1)$$

$D_{ij,kl}$ is the subalignment score. Θ_{global} is the pruning score used during global alignment. Θ_{local} is the local alignment pruning score. G_E is the gap elongation cost. l_I and l_K are the lengths of the subsequences being aligned, and L_I and L_K are the lengths of the sequences being aligned. Using this pruning scheme most global alignments between related sequences produce an alignment. Unfortunately, this also lowers the efficiency of the pruning significantly. Figure S1 shows the time needed to do an alignment without pruning divided by the time needed to do the same alignment using pruning as a function of the length difference between the two sequences. When the length difference is small, it is significantly faster to use pruning. When the length difference is large, there is only a small speed advantage. At a length difference of 25 the use of pruning is only ~20% faster than not using pruning.

The new implementation of the FOLDALIGN algorithm has a lower time and memory complexity than the old implementation during global alignment. Since a global alignment must include both ends of both sequences, the δ parameter can be used to also limit the start coordinate of a subalignment from the second sequence. In this way the δ parameter becomes similar to the M parameter used in [1]. The new time complexity is $O(L_{min}^3\delta^3)$ and the memory complexity is $O(L_{min}^2\delta^2)$, where $L_{min} = \min L_I, L_K$. The old implementation has a time complexity of $O(L_I^3 L_K \delta^2)$ and a memory complexity of $O(L_I^2 L_K \delta)$ since it used the local alignment algorithm with λ equal to the sequence lengths.

To train and test FOLDALIGN's global alignment performance a new dataset has been made. The sequence pairs of the dataset were selected from the 5S rRNA, RNaseP, SRP, and tRNA databases [2, 3, 4, 5]. Any sequences containing nucleotides other than *A, C, G* or *U* were removed from the databases. A few sequences which obviously did not fit into the databases, were removed. Then the sequences in each database were redundancy reduced to 90% similarity using the Hobohm 2 algorithm [6]. Sequence pairs were selected from the remaining sequences by sorting the pairs by their identity and selecting the pairs with the lowest identity. Each sequence can only be part of one sequence pair. The structures were cleaned by annotating any non *A - U, G - C,* or *G - U* base pair as single stranded. Nucleotides annotated to base pair with gaps were also reannotated to be single stranded.

The 5S rRNA database is split into three sections. Each section was treated separately before the final datasets were joined. This part of the data contains 215 sequence pairs. From the RNaseP database only the sequences in the bacterial type A alignment [7] were used as this alignment seems to have the most sequences and the best annotation. This dataset has 101 sequence pairs. The SRP dataset contains 121 sequence pairs. The pseudo knot base pairs were removed from the structures.

The tRNA dataset contains 1810 sequence pairs.

The 5S rRNA and tRNA datasets were used to select the parameters of the score matrix (gap penalties, substitution vs. energy weight, and Ribosum clustering percentage). The SRP and RNaseP datasets were used as validation datasets. As performance measure the Matthews correlation coefficient (MCC) of the base pair prediction was used [8]. Correctly predicted base pairs are counted as true positives, predicted base pairs which are not found in the annotation, are counted as false positives. Annotated base pairs not found in the prediction are counted as a false negative. Positions not predicted to base pair which are not annotated to base pair, are counted as true negatives. The average MMCs are: 5S rRNA 0.81, tRNA 0.86, RNaseP 0.50, and SRP 0.49. The results for the RNaseP and SRP datasets indicate that the good performances reported for 5S rRNA and tRNA may be due to over fitting. If this were the case, then it should also be possible to over fit on the RNaseP and SRP datasets. These datasets were therefore used to find alternative sets of parameters. The best MCC found for both datasets were 0.56. The poor performance therefore does not seem to be due to over fitting. Some of the performance difference is likely to be due to structural inserts in the structures. Some of the sequence pairs in both the RNaseP and the SRP datasets contain stem inserts which FOLDALIGN currently can not handle. The 5S rRNA and tRNA datasets contain fewer large stem inserts.

Recently a paper was published [9] which showed that while FOLDALIGN makes good alignments its base pair prediction sensitivity is lower than that of other methods for folding and aligning RNA sequences. Figure S2 shows the performance of the old and new versions of FOLDALIGN using the data set from [9]. Comparing Figure S2 and Figure 7B in [9] shows that the new version is as good as that of other methods. The "2.1. Clean" curve shows the base pair sensitivity when any non *A - U, G - C,* or G - U base pairs have been removed from the structure annotation. Whether these extra base pairs are correct base pairs or artifacts from the use of predicted alignments in RFAM [10] remains to be clarified.

In [9] the time and memory needed to align two sequence pairs for several different methods are compared. The two examples are: tRNA (RD0260 vs. RE6781, 77 nt. vs. 76 nt.), and 5S rRNA (M16172 vs. X02128, 117 nt. vs. 116 nt.). Using FOLDALIGN version 2.0 it takes 19 and 102 seconds to align the two pairs. The alignments need 82 mb and 282 mb of memory. Using the new version of the algorithm aligning the examples takes 2 and 6 seconds. The maximum amounts of memory needed are 8 and 11 mb. The machine used to make these tests runs Linux (kernel 2.6) on two 2.4 GHz 32 bits Intel Xeon CPUs. The machine has 4 gigabytes of memory.

References

[1] Mathews, D & Turner, D. (2002) *J Mol Biol.* **317**, 191–203.

[2] Szymanski, M, Barciszewska, M, Erdmann, V, & Barciszewski, J. (2002) *Nucleic Acids Res.* **30**, 176–8.

[3] Brown, J. (1999) *Nucleic Acids Res.* **27**, 314.

[4] Rosenblad, M, Gorodkin, J, Knudsen, B, Zwieb, C, & Samuelsson, T. (2003) *Nucleic Acids Res.* **31**, 363–4.

[5] Sprinzl, M & Vassilenko, K. (2005) *Nucleic Acids Res.* **33**, D139–40.

[6] Hobohm, U, Scharf, M, Schneider, R, & Sander, C. (1992) *Protein Sci.* **1**, 409–17.

[7] Harris, J, Haas, E, Williams, D, Frank, D, & Brown, J. (2001) *RNA.* **7**, 220–32.

[8] Matthews, B. (1975) *Biochim Biophys Acta* **405**, 442–451.

[9] Dowell, R & Eddy, S. (2006) *BMC Bioinformatics.* **7**, 400.

[10] Griffiths Jones, S, Moxon, S, Marshall, M, Khanna, A, Eddy, S, & Bateman, A. (2005) *Nucleic Acids Res.* **33**, D121–4.

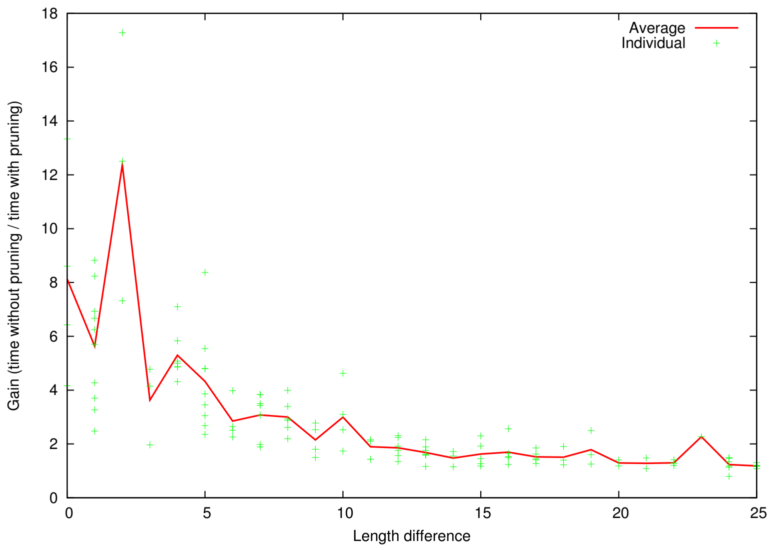


Figure S1: The alignment time without pruning divided by the alignment time with pruning as a function of the length difference between the input sequences. The SRP dataset and $\delta = 25$ was used. The curve shows the average gain. The points are the individual measurements

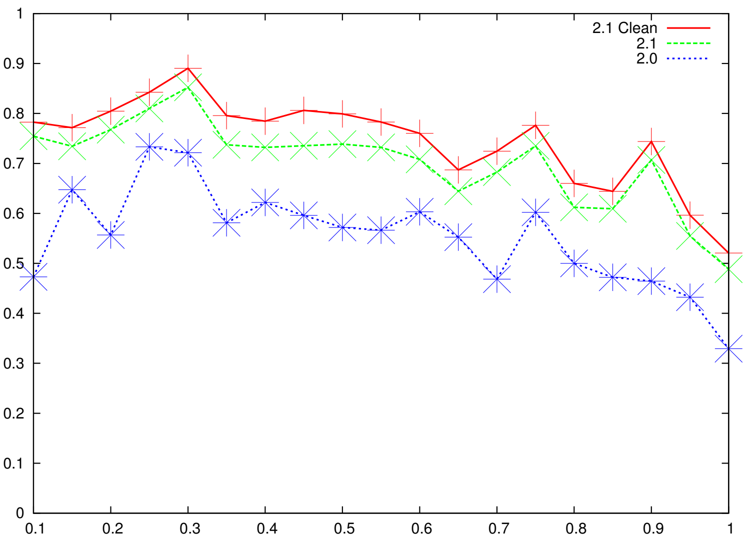


Figure S2: The base pair sensitivity as a function of sequence identity. For description of the data and method see [9]. Note that the scale of the sensitivity axis has been changed compared to Figure 7B in [9]. “2.1” is the new implementation with the new default parameters for global alignment. “2.1 Clean” also uses the new implementation but any non $A - U, G - C$, or $G - U$ base pairs have been removed from the annotation before measuring the sensitivity. The “2.0” curve is shows the results using the old 2.0 implementation with its global parameters. This curve is the same as the one in [9]

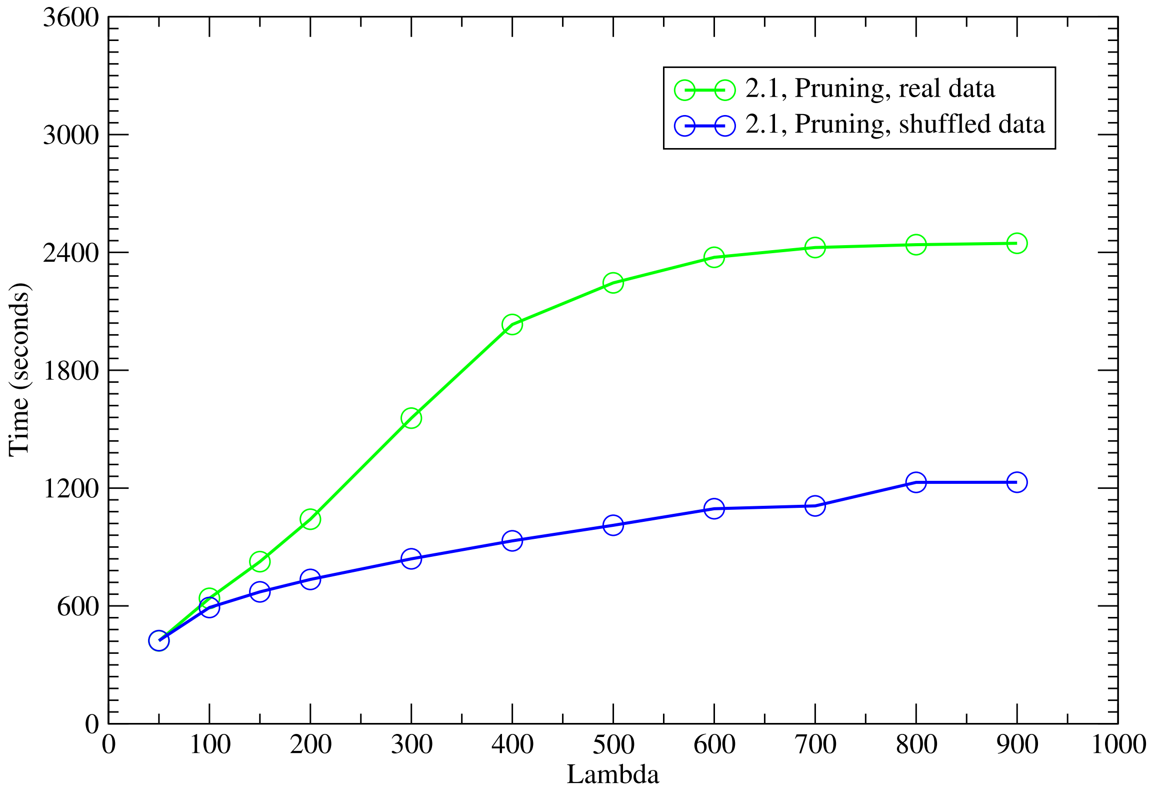


Figure S3: Average run time as a function of λ. The curves in this figure are the same as the "2.1, Pruning" curves in Figure 1. See Figure 1 for details

Torarinsson et al. [2007]: Multiple structural alignment and clustering of RNA sequences

Torarinsson et al. [2006]:
Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure

Torarinsson et al. [2006]: Erratum

Erratum

Genome Research 16: 885–889 (2006)

Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure

Elfar Torarinsson, Milena Sawera, Jakob H. Havgaard, Merete Fredholm, and Jan Gorodkin

The authors misunderstood the notation used by MultiZ alignments, which unlike BLAST and many others, does not represent all positions relative to the leading strand. Therefore, they did not scan pairs adjacent to +/+ and +/− alignments; instead, the positions of alignments are relative to the 5′ end of the strand in question. This misunderstanding led them to missing unalignable regions in the vicinity of +/− alignments. The authors note that all the analyses are still correct, and that they're only analyzing the 36,970 pairs adjacent to +/+ alignments, and not the additional 18,956 pairs that are adjacent to a +/− alignment. Also, there are not ~100,000 pairs but ~185,000 pairs altogether between the two genomes. The authors are now in the process of scanning these regions along with the rest of the chromosomes.

Gorodkin et al. [2006]: MicroRNA sequence motifs reveal asymmetry between the stem arms

Gorodkin et al. [2006]: Supplementary material

Supplementary material:

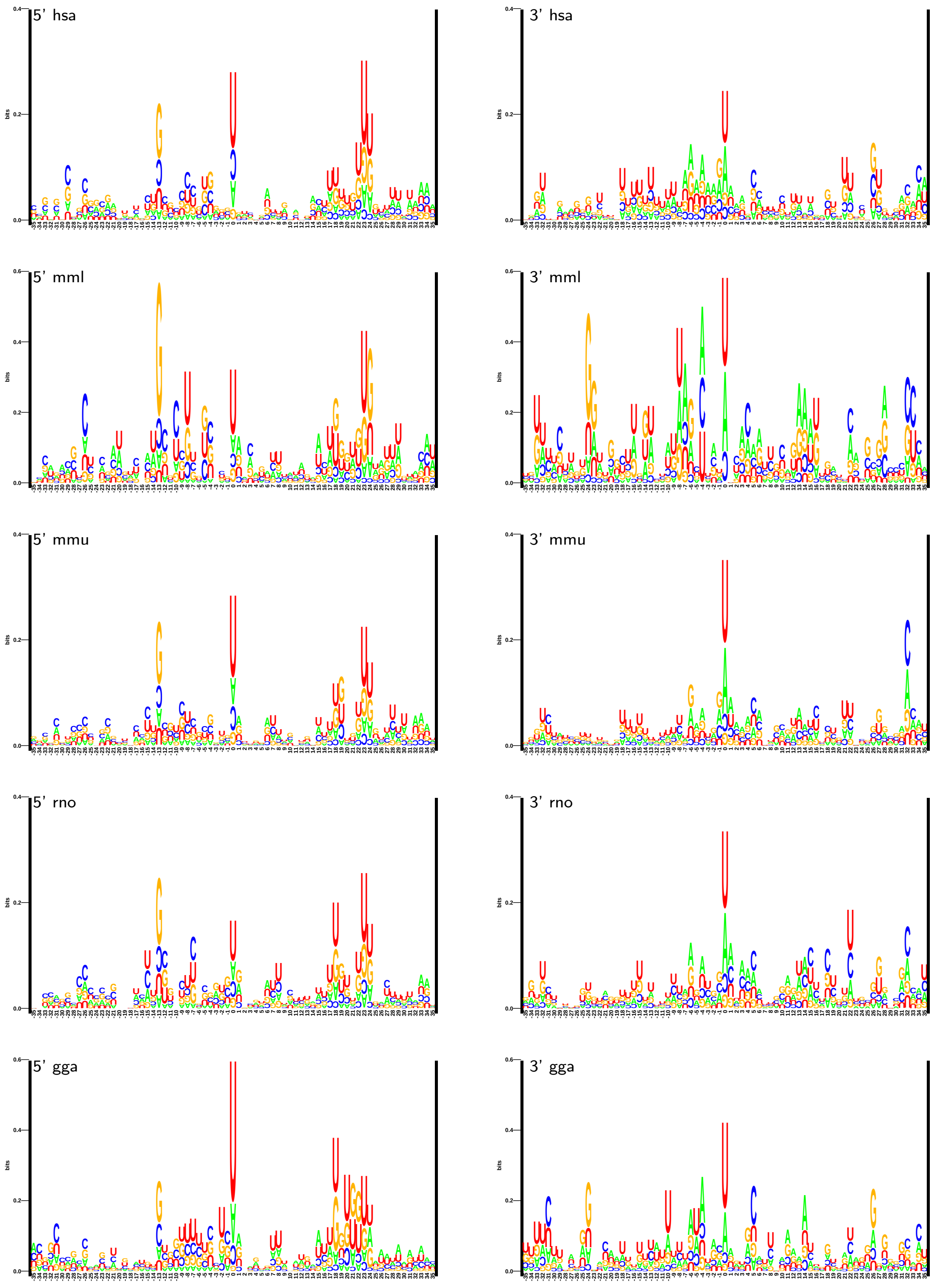
MicroRNA sequence motifs reveal asymmetry between the stem arms

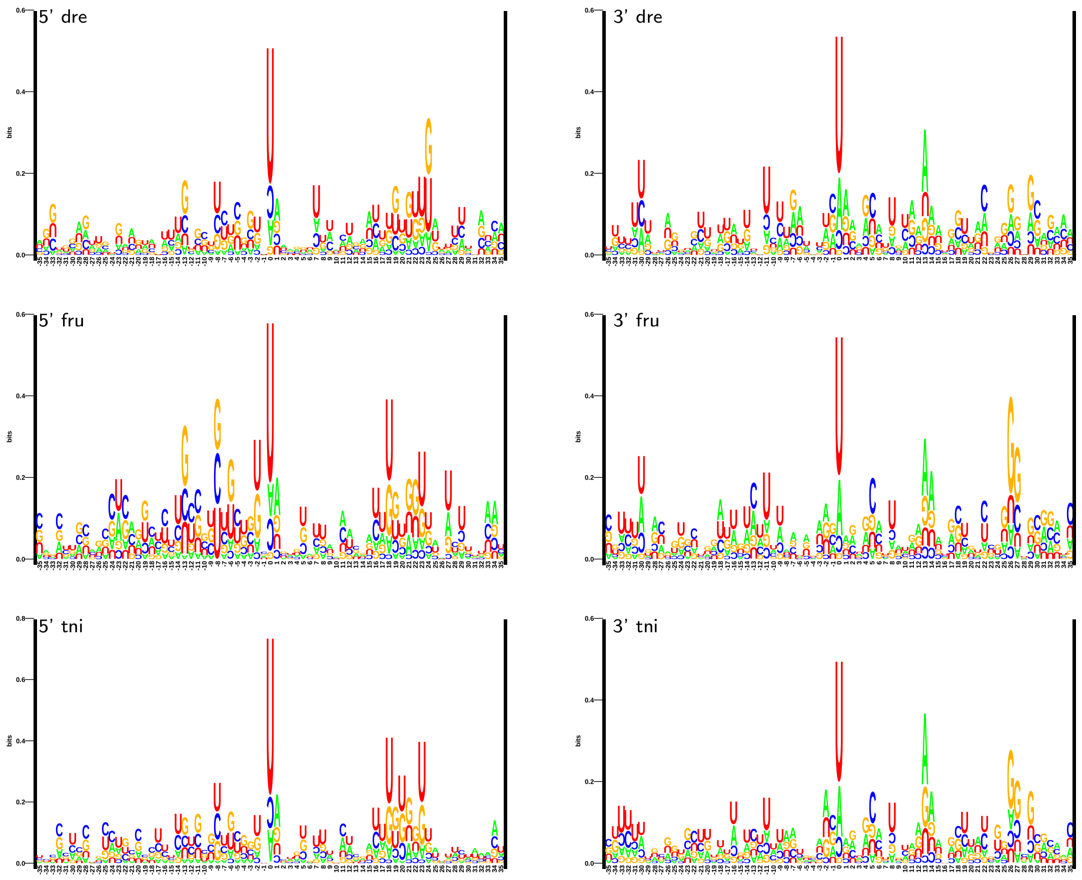
Jan Gorodkin[1], Jakob H. Havgaard[1], Mats Ensterö[2], Milena Sawera[1], Peter Jensen[1], Marie Öhman[2] and Merete Fredholm[1]

[1]Division of Genetics and Bioinformatics, IBHV and Center for Bioinformatics, The Royal Veterinary and Agricultural University, Grønnegårdsvej 3, DK-1870 Frederiksberg C, Denmark
[2]Department of Molecular Biology & Functional Genomics, University of Stockholm, SE-106 91 Stockholm, Sweden

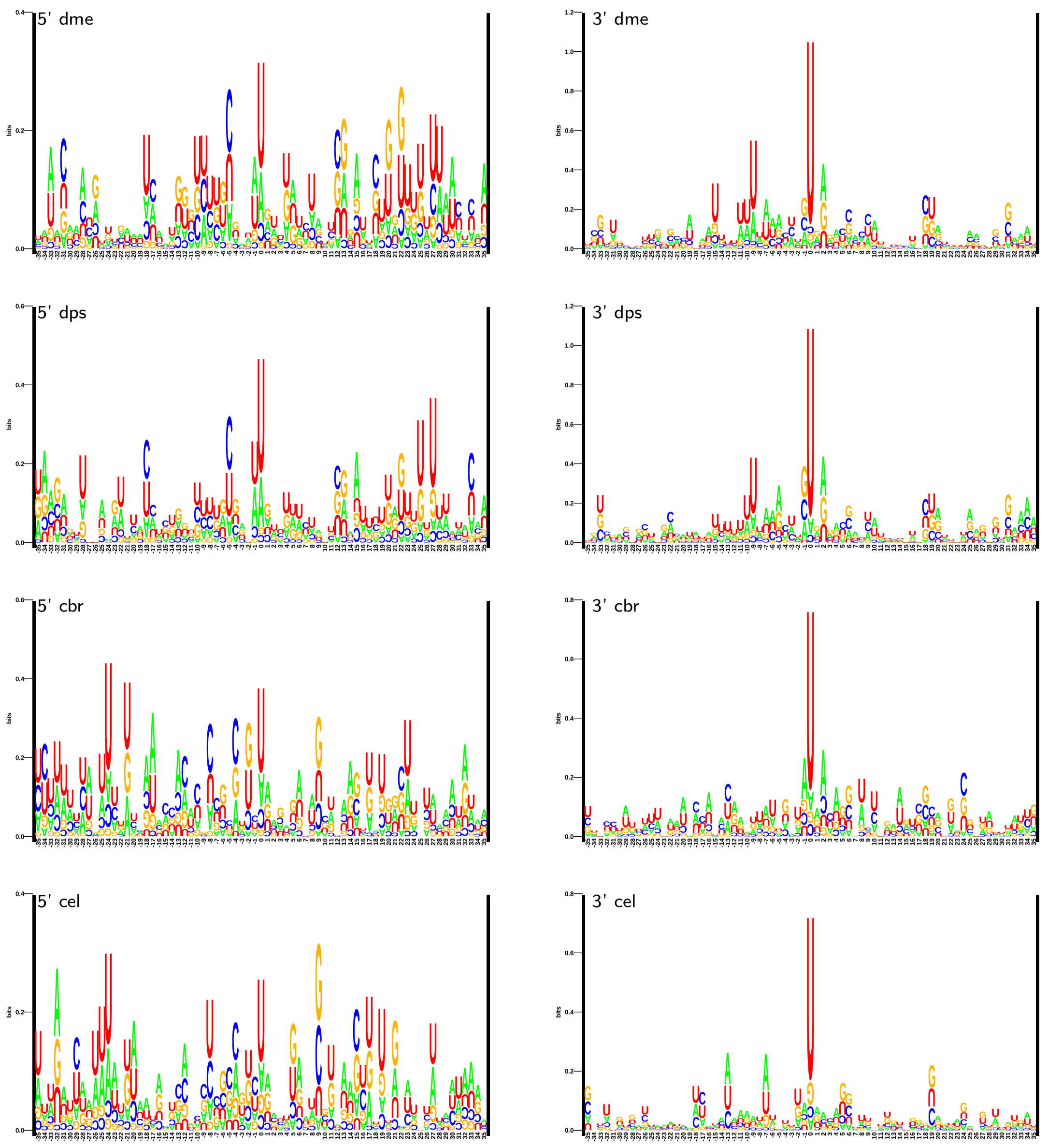
1 Sequence profiles of 5’ and 3’ arms

Figure S1: The sequence profiles various organisms. Left column represent 5’ arm motifs. Right column represent 3’ arm motifs. The scales are arbitray.

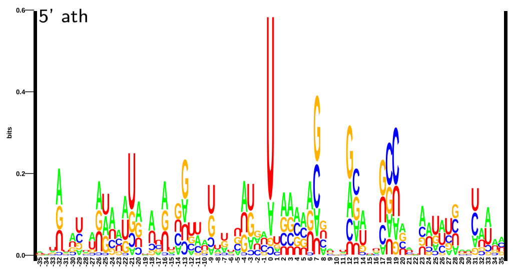


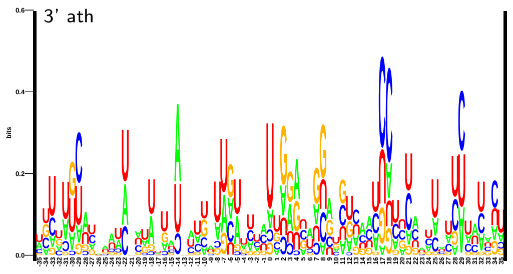


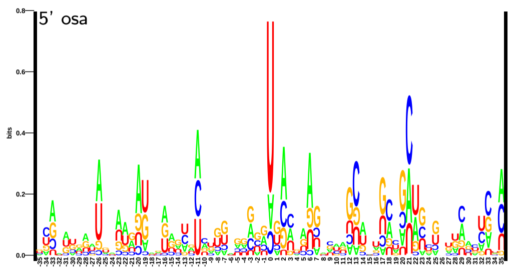
3

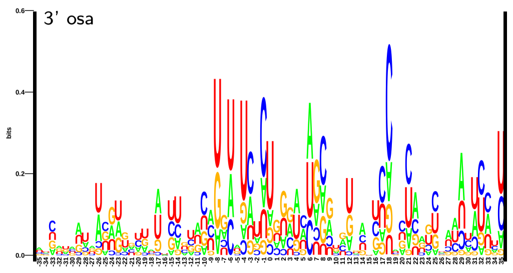


4


5' ath
bits


3' ath
bits


5' osa
bits


3' osa
bits