RNA Homology Searches Using Pair Seeding

by

Sriram Darbha

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2005

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Due to increasing numbers of non-coding RNA (ncRNA) being discovered recently, there is interest in identifying homologs of a given structured RNA sequence. Exhaustive homology searching for structured RNA molecules using covariance models is infeasible on genome-length sequences. Hence, heuristic methods are employed, but they largely ignore structural information in the query. We present a novel method, which uses secondary structure information, to perform homology searches for a structured RNA molecule. We define the concept of a *pair seed* and theoretically model alignments of random and related paired regions to compute expected sensitivity and specificity. We show that our method gives theoretical gains in sensitivity and specificity compared to a BLAST-based heuristic approach. We provide experimental verification of this gain.

We also show that pair seeds can be effectively combined with the spaced seeds approach to nucleotide homology search. The hybrid search method has theoretical specificity superior to that of the BLAST seed. We provide experimental evaluation of our hypotheses. Finally, we note that our method is easily modified to process pseudo-knotted regions in the query, something outside the scope of covariance model based methods.

Acknowledgments

I extend my sincere thanks to my supervisor Prof. Ming Li for his guidance throughout my research. It was during his graduate course in Bioinformatics Algorithms that I came up with the idea for my research topic. I would also like to express my sincere gratitude to my co-supervisor Prof. Daniel Brown, who generously gave many hours to mentor me, well before I was officially his student. Prof. Brown's graduate course on Genome Sequence Analysis was illuminating and forms the foundation of my knowledge of this field.

I thank Prof. Therese Biedl for giving her time to be a reader for my thesis, even though she was not in town at the time of my thesis submission. I especially enjoyed her graduate course on Graph Theory. I would also like to thank Prof. Forbes Burkowski for agreeing to be a reader for my thesis, particularly given his knowledge of algorithms for structure prediction of RNA molecules. I learnt a lot about RNA in his graduate course on Structural Bioinformatics.

Thanks are also due to my office-mate Richard Jang and fellow graduate student Thomas Tang. I found technical discussions with them to be enjoyable as well as fruitful, and hope that I have enriched their graduate student experience as much as they have mine.

Finally, I thank NSERC for their generous support through the PGS-M scholarship, and the University of Waterloo for Teaching Assistantships, the Graduate Incentive Award and the Faculty of Mathematics Graduate Scholarship, all of which provided vital financial support to my family and myself.

Dedication

I would like to dedicate this work to my daughter Aparna, whose presence reminds me each day that the best things in life are free. This dedication seems fitting for another reason – her name has in it, I realized only recently, the term "rna"!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Homology searching is one of the most basic tasks in the field of computational biology. Given a *target* sequence and a *query* sequence, homology searching refers to finding "close" matches to the given query in the target. This is useful in numerous contexts for researchers in the life sciences. For example, having identified and sequenced a novel murine gene, a researcher could use a homology search tool to discover functionally related genes in the genome of another mammal, such as the human.

Evolution proceeds so as to conserve the function of biological molecules across related species. For example, the sequences of genes that code for the Histone H4 proteins (which package DNA into condensed form during cell division) are strongly conserved across diverse eukaryotic species from the pea plant to the cow, indicating how crucial those genes are for the viability of these organisms [10]. In contrast, a DNA sequence that does not code for useful elements experiences no such selective pressure across related species, and eventually diverges over time. Like protein-coding DNA, a functionally active non-coding RNA (ncRNA) molecule experiences selective pressure to conserve its function across species. However, unlike DNA, ncRNA molecules often exhibit characteristic secondary structures, resulting from the pairings of complementary bases within their sequences. For example, the bases A and U form a complementary pair, as do bases G and C. These pairings are called *Watson-Crick* (W-C) pairings, after James Watson and Francis Crick who discovered the structure of DNA. In addition, RNA also contains G-U pairings, which are weaker than the W-C ones. In this work, we refer to the 4 W-C pairings, along with G-U and U-G as *relevant* pairings. The remaining 10 pairings, such as G-G, A-C etc., are rare in RNA molecules.

Traditional homology search methods, such as BLAST [1] [1], have been designed to look only at conservation patterns in the primary sequence, *i.e.*, the sequence of nucleotides. This is a satisfactory approach when searching for homologs of generic DNA and protein-coding genes, since a lot of the information content in such sequences is in the primary sequence alone. However, there is growing evidence that ncRNA genes conserve secondary structure more than primary sequence [8]. That is, they carry information as part of their structures, in addition to primary sequence. Since search techniques based on primary sequence alone ignore this structural information, they can potentially miss distant homologs when the query is a structured RNA sequence.

To address this deficiency, homology search methods based on *covariance models* were developed to allow exhaustive searches of homologs of structured RNA molecules [7]. These methods are computationally intensive, requiring cubic time in the size of the model, and are thus too slow to be executed on larger genomic sequences. This has resulted in the development of heuristic approaches for RNA homology [9, 15]. However, much as for the previously existing heuristics like BLAST, these heuristic methods ignore crucial secondary structural information in the query RNA sequence.

In this thesis, we present a prototype heuristic method to find homologs of a single structured RNA sequence. This problem is related to, though different from, searching for homologs of an RNA *family*. We define the concept of a *pair seed* and use it to develop an efficient algorithm that incorporates structural information into the search technique. The rest of this thesis is organized as follows: Section 2 reviews the background information for this work, starting with the concept of *seeding* used in traditional homology search methods. We briefly review recent work to find homologs of a single structured RNA molecule. Section 3 introduces our idea of a pair seed and motivates its application to the area of RNA homology searching. We demonstrate the theoretical superiority of our approach via calculations of expected sensitivity and specificity. Section 4 presents the algorithms we developed that comprise our homology search technique. Section 5 summarizes the experiments we performed to evaluate our approach, and concludes with a discussion of our results and future work in this area.

[1] The term BLAST refers in this thesis to the BLASTN method to perform nucleotide-based searches.

Chapter 2

Review of Background Material

2.1 Seeding in Homology Searching

Homology searching refers to finding close matches to a given query sequence in a target sequence database. For nucleotide sequence alignments, the quality of an alignment is often measured simply by scoring matching nucleotides as $+1$, mismatches ones as -1 and using affine penalties to score gaps. For protein sequence alignments, closeness is typically measured using pairwise scoring matrices such as BLOSUM [11] and PAM [5].

BLAST [1] and 2-hit BLAST [2] are among the most popular homology search methods currently in use. Both methods are based on a "seeding" approach that employs two steps: in the first step, so-called *seed* hits are identified. These are ungapped regions in the query and target that match each other. In the second step, seed hits are extended on either side, to find high-scoring regions of homology. BLASTN requires a single seed hit of size 11. In contrast, 2-hit BLASTN requires *two* seed hits – with the required length of each being smaller than for a single BLASTN seed hit. The two hits are required to have no net insertions or deletions in the sequence between them – in other words, they are *on the same diagonal* of the dynamic programming matrix for the two sequences. All pairs of seeds that satisfy these criteria are extended in the second stage. BLASTN identifies seed hits relatively fast, while spending a larger fraction of the time extending them into potential homologous regions; 2-hit BLASTN, on the other hand, spends more time identifying pairs of seeds to extend, spending relatively less time in the second stage extending those hits [14].

Seed length has direct implications on sensitivity and specificity of the homology method. Sensitivity is a measure of how many of the total homologs of the query are identified in the target; 100% sensitivity is the ideal. Specificity is a measure of how many of the identified matches are real homologs. Again, as high a specificity as possible is desirable. Increasing the seed size in the first step results in fewer seed hits to extend in the second step, and consequently high specificity. However, increasing seed size means fewer true hits because many alignments do not include a hit, which results in lower sensitivity. Conversely, decreasing the seed size results in higher sensitivity at the expense of specificity.

This balance between sensitivity and specificity is common when attempting to optimize a seeding method's performance. We show here that, under certain assumptions, it is possible to seed RNA homology searches so that we achieve gains in both sensitivity and specificity, as compared to a BLAST-based approach.

2.2 Spaced and Vector Seeds

Recently, Ma *et al.* [13] extended the concept of seeds by inventing *spaced* seeds. A spaced seed is composed on 1's and 0's, the 1's identifying positions which require a match and the 0's identifying "don't care" positions that do not have to match, but could. Using their terminology, an example of a length-18 seed that requires 11 matches is 11 1010 0101 0011 0111. For example, the sequences AAAAAAAAAAAAAAAAAA and AAAGAGGAGAGGAAGAAA would be considered hits to each other using the above seed, since they match in all the 1-positions of the seed. With this notation, the length-11 BLASTN seed is simply 111 1111 1111 since it requires 11 consective nucleotides to match, for a hit to be recorded. Note than the two sequences given above would not be considered hits to each other using the BLASTN seed.

Ma *et al.* showed that spaced seeds deliver a significant gain in sensitivity compared to the contiguous seed used by BLASTN. Using a model of related sequence that treats each position as an independent Bernoulli trial with a fixed homology level, Ma *et al.* showed that the optimal spaced seed is 50% more sensitive that the default BLASTN seed (with comparable specificity), assuming a probability of 0.7 for individual nucleotide conservation. Subsequently, Brejová *et al.* [4] developed a protein-coding region model of sequence, and showed that the predicted and empirical sensitivity of spaced seeds optimized under that model are provably better than those of the BLASTN seed.

More recently, the YASS [14] algorithm generalized these concepts by allowing multiple, possibly overlapping, seeds to be "chained" together to form a hit that is then extended.

Later, Brejová *et al.* developed *vector* seeds [3] to extend the idea of seeding to protein homology searches. A vector seed, like a spaced seed, is composed of a seed template of 1's and 0's. In addition, it also has an associated threshold score, based on a given scoring matrix *e.g.* BLOSUM-62. For example, the vector seed (1101, 13) would score the alignment of two 4-residue peptide sequences as the sum of the alignment scores of residues 1, 2 and 4; the alignment would be recorded as a hit if the total score is 13 or more. Vector seeds are more expressive than spaced seeds (any spaced seed can be represented as a vector seed). Vector seeds have been shown to result in increased sensitivity and specificity in protein homology searching using BLOSUM scoring matrices.

Pair seeds, which we introduce in this work, are similar to vector seeds in that they are associated with a scoring matrix and a threshold score. However, each position in a pair seed represents two nucleotides, whereas in nucleotide seeds it is only one.

2.3 RNA Homology and Covariance Models

Recently, there are has been increased interest in identifying homologs of non-coding RNA (ncRNA) genes, which code for RNA products that are not translated to protein. The number of ncRNA genes discovered has been increasing rapidly [8]. These sequences are different from other nucleotide sequences in a crucial way: their nucleotides can internally bond. This internal pairing of an RNA molecule gives it a characteristic "secondary" structure, which is made up of long-range, variable length complementary pairing between positions in the sequence. An example is given in Figure 2.1, and the corresponding secondary structure is shown in Figure 2.2.

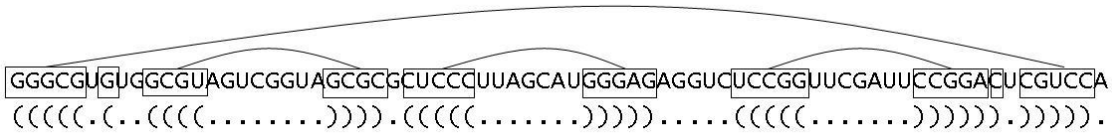

Figure 2.1: Sequence and structure of the *S. cerevisiae* Alanine tRna [12]

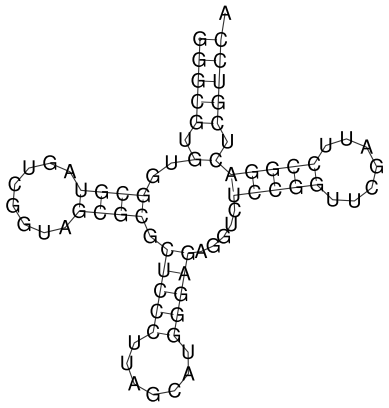

Figure 2.2: Secondary structure of the *S. cerevisiae* Alanine tRNA

Hidden Markov models (HMM) have been employed previously in homology searching [6]. However, HMMs cannot capture nested correlations of variable separation, since their topology restricts them to processing sequences in a left-to-right linear manner. Hence, they are not well-suited for capturing secondary structure information in ncRNA molecules.

To this end, Eddy developed covariance models (CM) to capture features of a set of structured RNA molecules [7]. Covariance models are based on stochastic context free grammars (SCFG). For protein and nucleotide sequences, profile HMMs [6] are used to model characteristics of a set of related sequences, such as a family of genes. Covariance models can be thought of as more complex versions of profile HMMs, in that they capture nested dependencies, by modeling the sequence as a set of segments built recursively from the outside in. We do not consider covariance models in any more detail in this thesis.

It is known for HMMs that the Viterbi and forward algorithms are able to identify the optimal parse of a given sequence through a given HMM [6]. These algorithms execute in linear time in the size of the HMM. However, due to the additional complexity of a CM, the corresponding parsing algorithm for CMs, the forward algorithm, requires cubic time in the size of the query [6]. Durbin *et al.* point out that if the maximum match size is limited to D nucleotides, the runtime can be written as $O(MD^2L)$, where M is the model size and L is the database size. Regardless, this effectively limits the sizes of problems against which CM-based algorithms can be executed.

2.4 Finding Homologs of a Structured RNA Molecule

This work deals with the problem of identifying homologs of a single structured RNA molecule. While this problem is solvable by several approaches, there are some drawbacks to each method. We outline the known work in this area below.

2.4.1 The RSEARCH Algorithm and Ribosum Matrices

Klein and Eddy introduced the RSEARCH [12] program, the first to find homologs of a given structured RNA molecule. This is an exhaustive CM-based method that first builds a CM of a given query sequence and structure. It uses new scoring matrices [12] to score alignments of the query to the target sequence. Note that the aligned query and target sequences could contain unpaired as well as paired regions. The search is fully sensitive but suffers from the drawback of being very slow, as the underlying SCFG parsing algorithms require cubic time. In fact, the authors note that the approach is too slow to run on genome-scale sequences. One approach they propose is a multi-processor implementation.

The Ribosum family of matrices developed by Klein and Eddy allow scoring of alignments of two *nucleotide pairs*, in addition to alignments of two single nucleotides. We briefly outline the salient features of Ribosum matrices here. The discussion below borrows material heavily from their original paper [12], to which the interested reader is referred for an in depth discussion of how the matrices are generated.

There are a total of 170 matrices in the RSEARCH v1.0 release. Table 2.2 shows one matrix, the Ribosum-95. These matrices are developed, as per the authors, using a method that is analogous to that employed for generating the BLOSUM [11] matrices for protein sequence alignments. Specifically the method uses a multiple sequence alignment of a set of related RNA molecules. Ribosum matrices are names using the convention "RibosumXX-YY", where XX is the *percentage clustering* value and YY is a second parameter, the *percent identity* value. Pairs of sequences that have at least XX% identity are clustered together into groups. Pairs of sequences with less than YY% identity are not considered during the clustering operation. If not specified, the second parameter YY defaults to zero. For example, the Ribosum75 matrix has percentage clustering set to 75% and percent identity set to 0, so that all pairs of sequences are considered during clustering. A Ribosum matrix contains a 16x16 part for scoring alignments of nucleotide

pairs to pairs, and a 4x4 part to score alignments of single nucleotides, shown in Tables 2.1, 2.2. We come back to Ribosum matrices in Section 3.5.

Table 2.1: Ribosum-95 nucleotide alignment scoring matrix [12].

	A	C	G	U
A	1.96			
C	-1.33	0.99		
G	-0.91	-1.69	0.85	
U	-0.97	-0.62	-1.10	1.37

2.4.2 BLAST-based Heuristics

To get around the prohibitively slow speed of CM algorithms, heuristic approaches have been employed to speed up RNA homology searches. An example is the approach used by the RFAM database. RFAM [9] is a database of known ncRNA families and their member sequences. The database allows the searching of an unannotated genome for homologs of a given RNA family, where each RFAM family is represented by a corresponding CM. Each family's CM is built from a hand-curated multiple sequence alignment of all sequences that belong to that family. The problem of searching for homologs of a given RNA family is similar to the problem we consider in this work, that is, searching for homologs of a *single* structured RNA molecule. Annotation of a target with homologs of an RFAM family occurs in two stages: In the first stage, a BLAST search is executed with the nucleotide sequence of every member of a given RNA family, against the target database. All hits so found are used to demarcate "windows" of interest in the target sequence. In the second stage, the CM algorithm is executed only within the windows identified in the first step. The idea is to filter out most of the target, so that the exhaustive algorithm is only executed on a fraction of it.

There are two drawbacks of using a BLAST preprocessing step. Firstly, Ma *et al.* showed [13] that the all-1's seed used by BLAST has poor sensitivity, in comparison to other spaced seeds of the same weight. It follows that a substantial gain in sensitivity could be achieved through the use of optimized spaced seeds. Secondly, a nucleotide-only search essentially discards information related to long-range internal pairings in the sequence. As mentioned previously, the function of a ncRNA molecule is often tied to its secondary structure that results from pairings between complementary bases in the

Table 2.2: Ribosum-95 pair to pair alignment scoring matrix [12]. All positive scores are shown in boldface and correspond to alignments more likely in RNA than at random. The units are log-base-2 likelihood ratios.

	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
AA	-2.84															
AC	-8.54	-2.06														
AG	-9.17	-9.34	-1.33													
AU	-4.96	-1.63	-5.89	**4.22**												
CA	-9.97	-10.36	-10.35	-5.61	-6.19											
CC	-13.47	-8.79	-14.75	-2.48	-11.36	-3.69										
CG	-3.83	-5.35	-2.56	**2.06**	-3.14	-5.02	**5.22**									
CU	-11.94	-10.61	-9.50	-4.13	-8.34	-6.95	-4.33	-2.68								
GA	-6.95	-7.67	-9.63	-3.86	-7.99	-12.70	-5.85	-7.60	-1.65							
GC	-5.18	-2.70	-6.11	**2.89**	-5.34	-3.70	**2.82**	-2.20	-2.61	**5.50**						
GG	-9.66	-10.14	-4.36	-4.81	-11.46	-11.71	-4.66	-9.31	-9.51	-4.64	-2.19					
GU	-6.42	-4.68	-7.08	**0.74**	-6.09	-6.99	**0.26**	-4.45	-6.24	**1.43**	-5.52	**3.31**				
UA	-2.49	-5.39	-2.84	**1.90**	-2.90	-6.31	**2.96**	-4.32	-4.74	**1.94**	-6.22	**0.09**	**4.71**			
UC	-11.83	-8.91	-7.96	-4.30	-7.83	-8.56	-4.27	-4.58	-7.14	-4.06	-10.72	-4.57	-3.55	-3.99		
UG	-3.65	-7.12	-5.78	-0.37	-6.10	-8.23	**1.22**	-6.41	-7.63	**0.32**	-4.28	-1.33	**1.05**	-4.24	**3.18**	
UU	-8.89	-8.30	-9.54	-2.40	-8.87	-6.51	-3.04	-5.40	-9.79	-2.94	-5.03	-2.63	-1.59	-6.03	-3.68	-0.74

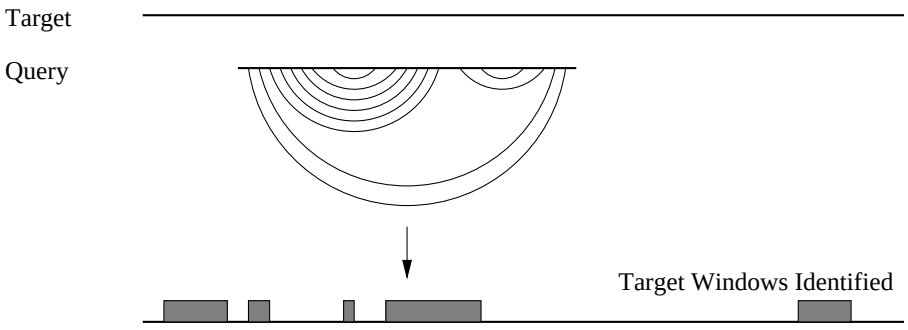

Figure 2.3: Two step heuristic search method of the RFAM database

molecule. The four bases A, C, G, U could pair to form the "Watson-Crick" pairs A–U and C–G. In addition, weaker pairings G–U are also seen frequently in RNA. Hence, an A–U pairing in an RNA molecule would be conserved over evolution if it mutates to a G–C pairing. Since such pairings are common and often form a significant portion of a ncRNA molecule, the nucleotide- only approach loses pairing information. Hence, distant homologs are likely to be missed by a BLAST search, if the RNA molecule has evolved in a such a manner that *pairings* between correlated positions have been conserved, even though the individual nucleotides have not been. From Ribosum-95 probabilities in Figure 3.2, it is seen that while Pr[nucleotide conserved] = 0.65 in paired regions, Pr[W-C pair conserved] = 0.76. Further, Pr[pair conserved] = 0.95, if we allow the pair to be a Watson-Crick pair or one of UG, GU. Hence, in general the chance that a nucleotide in a paired region mutates so as to conserve pairing is higher than the chance that it is itself conserved, between homologous RNA molecules.

2.4.3 The Hidden Markov Model Approach

Recently, Weinberg and Ruzzo [15] formulated a HMM-based approach for the problem of searching for homologs of a given RNA family. Their approach requires the existence of a CM for the given family, and knowledge of a *family-specific* threshold score. Starting with the CM, they discard pairing information and distill the remaining model into a profile HMM. Despite throwing away useful information, they employ a novel mechanism to parameterize their HMM, so that any sequence whose CM parse scores above the threshold is guaranteed to have a HMM parse that scores above a corresponding threshold.

Thus, their method benefits from the speed of the Viterbi algorithm for HMMs while

maintaining 100% sensitivity. However, the method does have some drawbacks as applied to single structured RNA molecules. Firstly, it relies on the existence of an accurate, hand-curated CM for a given family, and a family-specific threshold score. Hence it is currently unable to search for homologs of a single RNA sequence whose family is not known. It may be likely that one could get around this limitation by first using the RSEARCH [12] method to build a "general" CM for the given RNA sequence, and using that CM as input to Weinberg's approach. However, it is not clear how the *family-specific* threshold score would be determined for the CM auto-generated by RSEARCH. This score is necessary in parameterizing the profile HMM and delivering efficiency in database filtering. Further, and perhaps more importantly, it is not clear if a general CM auto-generated by RSEARCH is comparable in sensitivity to one built from a hand-curated multiple sequence alignment. Finally, their algorithm is not able to handle pseudo-knotted structures, so that any information regarding pseudo-knotted paired regions in the query is simply ignored. In pointing out this limitation, the authors claim that "studies suggest that pseudoknots contain little information" [15].

The pair seeding approach we have developed is easy to apply to a single structured RNA sequence. With a slight runtime penalty, our algorithm successfully hashes locations of paired regions, in expected linear time. This is significant, since both the BLAST preprocessing and the profile HMM methods discard pairing information during searches. Further, it is an interesting and unintended consequence that, with minimal modifications, our method can incorporate pseudo-knotted paired regions into the search algorithm (we describe later how to do this). A drawback of our approach is that its applicability to RNA molecules with fragmented paired regions is limited.

Chapter 3

Motivating Idea and Key Concepts

In this section, we define the problem of finding homologs of a single structured RNA sequence, and then present the key ideas behind our proposed solution.

3.1 Problem Definition

We are given an unannotated target nucleotide sequence $t = t_1 \dots t_n$. We are also provided a query RNA sequence $q = q_1 \dots q_m$ and its corresponding secondary structure, where pairings between nucleotides in the query are denoted as pairs of indices (l, r), where l is less than r. We can define an ungapped alignment between a query substring $q_i \dots q_j$ and a target substring $t_k \dots t_{k+j-i}$, such that for all pairings (l, r) in the query, either both l and r are within the range $i \dots j$, or neither of them is. The score of such an alignment is defined as Score $= \sum_{unpaired\ l \in i \dots j} M(q_l, t_{l+k-i}) + \sum_{pairing\ (l,r),\ l,r \in i \dots j} M(q_l q_r, t_{l+k-i} t_{r+k-i})$, given the scoring matrix M to score alignments of nucleotides and pairings. The first summation considers unpaired nucleotides and the second deals with pairings of nucleotides.

We are given a nucleotide seed s^N defined exactly as per Ma *et al.* [13]. We define a pair seed s^P similarly, having length L and weight W, as $s^P = s^P_1 \dots s^P_L$, where $s^P_i = 0$ or 1, $i = 1 \dots L$. Ma *et al.* have applied spaced seeds to ungapped nucleotide alignments. Our key idea is to use short ungapped alignments of paired regions as seeds. An ungapped paired region in the query of length L with outermost

pairing (l, r) consists of the two subsequences $q_l \ldots q_{l+k-1}$ and $q_{r-k+1} \ldots q_r$. The alignment of these query subsequences to the target, such that the query pairing (l, r) is aligned to the hypothetical pairing (a, b) in the target, is scored using the pair seed s^P as $Score = \sum_{i=1}^{L}(s_i^P.M(q_{l+i-1}q_{r-i+1}, t_{a+i-1}t_{b-i+1}))$. A seed hit is one that scores above a given threshold score T.

3.2 Motivating Idea

Our key idea is to model paired regions in the query RNA as sequences of *pair symbols*, and to use short ungapped paired regions as seeds.

For example, consider the query sequence in Figure 3.1. A nucleotide based homology method would simply treat this as the sequence GGGCCUU ... AAGGCUC, ignoring the internal pairing in the query. However, by noting the internal pairing, we can say that there is a "pair 7-mer" GC-GU-GC-CG-CG-UA-UA in the sequence.

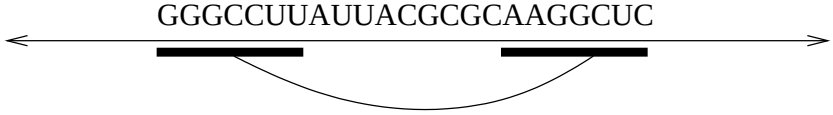

Figure 3.1: A RNA query sequence with a single paired region.

Let us say that the third pair GC in this query has mutated to an AU in a related RNA in a different species. A traditional homology method trying to find hits to the sequence in Figure 3.1 would note mismatched nucleotides in position 3 and 21 (in GG**G**CCUU versus GG**A**CCUU, and a second one in AAGG**C**UC versus AAGG**U**UC). However, an alignment method that knows that G and C in the query are paired could score GC against AU, and recognize that the two nucleotide mismatches actually conserve the complementarity of pairing. Since this is much more likely in RNA than in random sequence, such a method could distinguish homologs of RNA paired regions despite nucleotide mismatches.

To this concept of pair kmers, we add the well-known concepts of a *seed* for hashing, and of *hits* to query regions using the given seed.

3.3 Definitions of Key Concepts in Pair Seeding

Here we define key concepts necessary to present our proposed solution. We use the hypothetical example in Figure 3.2 to introduce the quantities being defined.

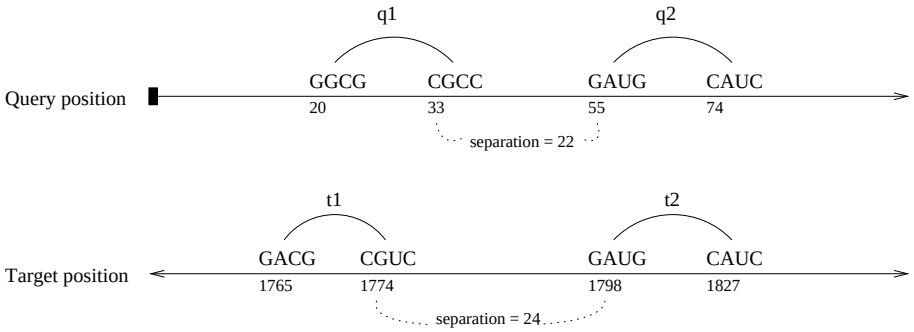

Figure 3.2: Motivating example for pair seeding concepts

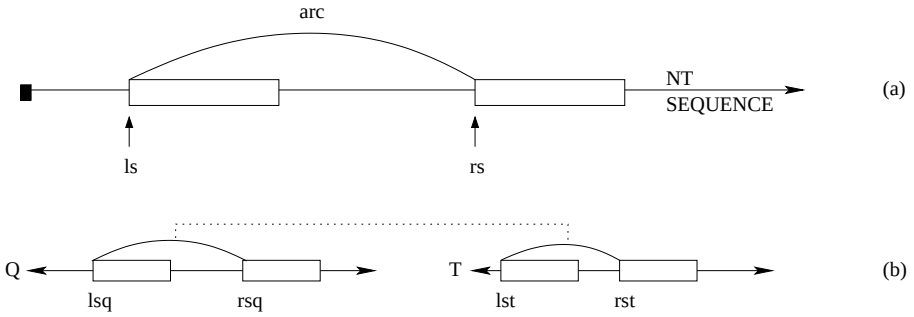

Figure 3.3: Key concepts: (a) a pair kmer, (b) a pair seed hit

Definition 1 *A* pair kmer *is defined as the ordered pair* $\mathbf{pk} = (\mathbf{ls}, \mathbf{rs})$*, where* **ls***(left start),* **rs***(right start) are positions in a nucleotide sequence. A pair kmer represents a sequence of k pair symbols. The k nucleotides starting at* **ls** *and those starting at* **rs** *are paired in complementary fashion, so that the nucleotide in position* $ls + i - 1$ *is paired with that in position* $rs + k - i$*, for* $i = 1, \ldots, k$*. Please see Figure 3.3(a).*

For example, Figure 3.2 shows the pair 4-mers $q_1 = (20, 33)$, $q_2 = (55, 74)$ in the query sequence, and the pair 4-mers $t_1 = (1765, 1774)$, $t_2 = (1798, 1827)$ in the target sequence. The 4-mers in terms of their pair symbols are q_1 = GC-GC-CG-GC, q_2 = GC-AU-UA-GC, t_1 = GC-AU-CG-GC and t_2 = GC-AU-UA-GC.

Definition 2 *The* **arc** *of a pair kmer is a positive integer value, defined as the distance between its left and right paired segments. That is,* $\mathbf{arc} = \mathbf{rs} - \mathbf{ls}$*, as illustrated in Figure 3.3(a).*

For example, in Figure 3.2, $\text{arc}(q_1) = 13$, $\text{arc}(q_2) = 19$, $\text{arc}(t_1) = 9$ and $\text{arc}(t_2) = 29$. Note that the arc of a pair kmer does not have an analogous quantity in ungapped nucleotide alignments.

Definition 3 *A* pair seed *is defined as a sequence of 1's and 0's, where a 1-position requires a match and a 0-position is a "don't care". A pair seed's* weight *is the number of 1's in it. A pair seed's* length *is the number of 1's and 0's. Note that this is exactly as per the definition of a spaced nucleotide seed [13].*

For example, the pair seed 10111 has weight 4 and length 5, while the ungapped pair seed 1111 has equal weight and length of 4.

Definition 4 *The* arc difference **ad** *of the alignment of one query and one target pair kmer is an integer value, defined as* $\mathbf{ad} = |arc_q - arc_t|$*, where* arc_q *and* arc_t *are arc values of the query and target kmers respectively.*

Consider Figure 3.2. For the alignment of q_1 to t_1, $\mathbf{ad} = |13 - 9| = 4$, while for the alignment of q_2 to t_2, $\mathbf{ad} = |29 - 19| = 10$. The arc difference of an alignment is loosely analogous to the concept of "diagonal" in nucleotide alignments, in the sense that both values quantify the degree to which insertions or deletions have occurred in the sequence bounded by the kmers. A diagonal shift of zero means that there have been no net insertions and deletions between two nucleotide hits. Similarly, an arc difference of zero for a pair kmer alignment means that the left and right segments of both kmers are separated by an equal number of nucleotides with no net insertions or deletions.

Definition 5 *The* arc margin **am** *for a homology search is the maximum allowed* arc difference *between a query and a target pair kmer, for the alignment to be considered a hit.*

An alignment between a pair kmer $Y = y_1 y_2 \dots y_n$ *in the query and another* $Z = z_1 z_2 \dots z_n$ *in the target is a pair kmer seed* hit *if its score* $\sum_{i=1}^{n} m_{y_i,z_i}$ *is equal to or*

exceeds the threshold T using the scoring matrix M, and $\mathbf{ad} \leq \mathbf{am}$*, where* **ad** *is the* arc difference *of the alignment and* **am** *is the* arc margin *for the homology search.*

A pair seed hit $\mathbf{h}$ *is defined as a tuple* $\mathbf{h} = (\mathbf{ls_q}, \mathbf{rs_q}, \mathbf{ls_t}, \mathbf{rs_t}, \mathbf{ps})$ *which uniquely identifies the left and right segments of the two pair kmers aligned to each other and the pair seed model used, as illustrated in Figure 3.3(b).*

Consider Figure 3.2, where q_2 is a perfect match to t_2 in terms of their pair symbols. However, if we use $\mathbf{am} = 5$, t_2 would not be considered a hit to q_2 since their arc difference 10 exceeds the arc margin.

Consider Figure 3.2 and the Ribosum-95 scoring matrix. The alignment of q_1 and t_1 scores 19.11 while that of q_2 and t_2 scores 19.65. For $T \leq 19.11$ and $\mathbf{am} = 10$, both alignments would be considered hits. For $T \leq 19.10$ and $\mathbf{am} = 5$, only the first would be considered a hit, even though the second one scores higher, as it has too large an arc difference. For $T = 19.6$ and $\mathbf{am} = 10$, the latter alignment would quality as a hit, while the former would not, as it scores too low. Finally, for $T = 19.6$ and $\mathbf{am} = 5$, neither alignment would be a hit. These examples show that variations in the arc margin and threshold can result in different hits being identified by the homology method.

The quantities defined above relate to a 1-hit method of homology searching, that is, where a single pair kmer match forms a seed hit for extension by use of a CM algorithm. We may also use a 2-hit model, where two seed hits are chained together to form an extensible hit, we define the additional quantities below.

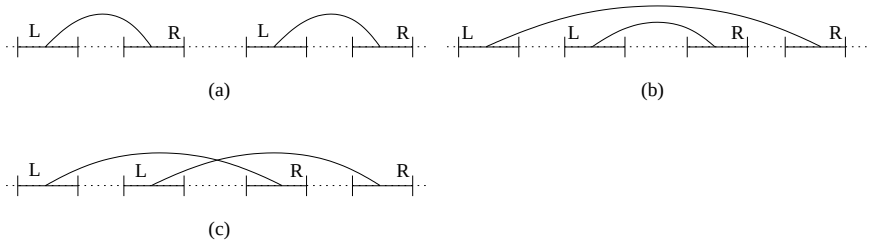

Figure 3.4: Separation of two pair kmers. Each pair kmer is shown with its left (L) and right (R) paired regions. There are three possible orientations of the two pair kmers: they can be (a) separate, (b) nested one within the other, (c) in a "pseudo-knotted" configuration.

Definition 6 *The* separation **sep** *of two pair kmers pk_1, pk_2 is an integer defined as:*

$$sep(pk_1, pk_2) = \begin{cases} ls_{pk_2} - rs_{pk_1}, & \text{if } pk_1, pk_2 \text{ are separate, as in Fig. 3.4(a)} \\ ls_{pk_2} - ls_{pk_1}, & \text{if } pk_1, pk_2 \text{ are nested, as in Fig. 3.4(b)} \end{cases}$$

The separation is defined only for two kmers that are disjoint, that is, without overlaps of their paired segments. Assuming they are disjoint, two kmers can either be nested one inside the other, occur one after the other, or be pseudo-knotted. These three cases are shown in Figure 3.4. (Note that for each of these three cases, the left end of pk_1 can occur before or after that of pk_2, giving a total of six cases. The duplicate cases are not shown for the sake of brevity.) For the current implementation, we have chosen to define separation only for cases (a) and (b), disallowing the pseudo-knotted case, but it is trivial to extend this definition to the pseudo-knotted case if required. For the example in Figure 3.2, sep(q_1, q_2) = 22 and sep(t_1, t_2) = 24.

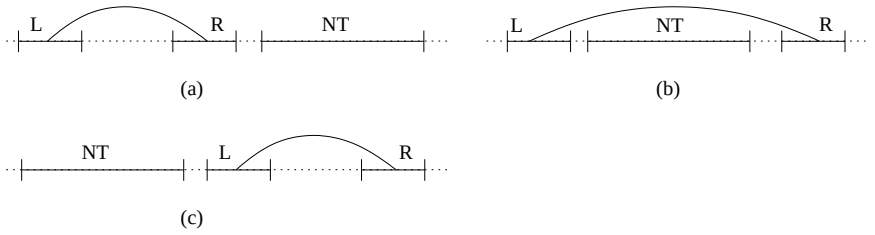

Figure 3.5: Separation of a Pair kmer and a Nucleotide lmer. The pair kmer is shown with its left (L) and right (R) paired regions. The nucleotide lmer is marked "NT". There are three possible orientations of the disjoint kmer and lmer: The nucleotide lmer is (a) after, (b) nested inside, (c) before the pair kmer.

Separation can also be defined for a pair kmer $pk = (ls, rs)$ and a nucleotide lmer. Note that a nucleotide lmer is defined as $ntl = (nt)$ since its starting position nt identifies it uniquely. This situation is shown in Figure 3.5, where the pair kmer has left and right segments of length k, while the nucleotide lmer is a single stretch of length l (possibly different from k). This definition is useful for a 2-hit algorithm which requires one hit to be a pair kmer match and the second hit to be a nucleotide match, each possibly using different seeds. We consider such a 2-hit algorithm in Section 4.4.3. In this case, we

differentiate between the cases when ntl occurs before, inside or after pk.

$$sep(pk, ntl) = \begin{cases} nt - rs, & \text{if } ntl \text{ is after } pk, \text{ as in Fig. 3.5(a)} \\ nt - ls, & \text{if } ntl \text{ is inside } pk, \text{ as in Fig. 3.5(b)} \\ ls - nt, & \text{if } ntl \text{ is before } pk, \text{ as in Fig. 3.5(c)} \end{cases}$$

Definition 7 *The* separation distance **sd** *of a pair of query kmers and a pair of target kmers is defined as the difference in separations of the query kmers and the target kmers. That is,* $\mathbf{sd}((q_1, q_2), (t_1, t_2)) = |sep(q_1, q_2) - sep(t_1, t_2)|$.

The separation distance is again analogous to the concept of "diagonal" in nucleotide alignments, just as the arc margin of 1-hit pair kmer alignments described above is. For example, in Figure 3.2, $\mathbf{sd}((q_1, q_2), (t_1, t_2)) = |22 - 24| = 2$. Intuitively, this quantity is a measure of how many net insertions and deletions have occurred between the pair of query kmers on one hand, and the pair of target kmers on the other.

Definition 8 *The* separation margin **sm** *of a 2-hit homology search method is a positive integer, and is the maximum allowed separation distance between two query pair kmers and their corresponding hits in the target, in order for the four pair kmers to be considered an extensible hit.*

Consider Figure 3.2 again. If a 2-hit homology search is performed with $\mathbf{sm} = 3$ (and **am**, T set so that t_1 is a hit to q_1, and t_2 is a hit to q_2) then the pair t_1–t_2 is a hit to the pair q_1–q_2. However, if $\mathbf{sm} = 1$, a hit is not recorded for these kmers.

3.4 Overview of Proposed Solution

In the last section, we defined the quantities necessary for describing our approach. In this section, we present an overview of our proposed solution to the problem of finding homologs of a single structured RNA molecule. Figure 3.6 summarizes the steps in our approach.

The first step is to choose a scoring matrix M, a pair seed s and a threshold score T. This is performed before the search algorithm executes. We describe in Section 3.5

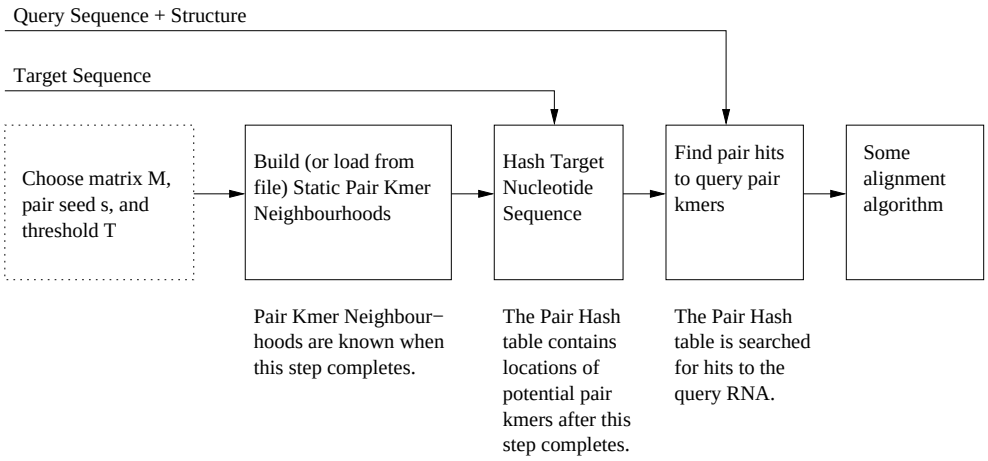

Figure 3.6: Overview of proposed solution. The solution consists of a first step in which parameter values are chosen. The three subsequent algorithmic steps together form the hit-finding phase of our homology search algorithm. The last step, performed by some alignment algorithm, involves extending seed hits to full alignments. The alignment algorithm for this step is independent of the algorithms for the three preceding steps. For a description of the steps, please see the text.

the rationale for choosing the Ribosum-95 matrix in our experiments. In Section 3.7, we outline why we choose pair 7-mers for 1-hit homology searches and pair 4-mers for the 2-hit method.

Using the Ribosum-95 matrix, the 7-mer x_1 = GC-GU-GC-CG-CG-UA-UA scores 34.17 when aligned to itself, and 29.86 when aligned to x_2 = GC-GU-**AU**-CG-CG-UA-**CG** (third and last pairs mutated). These are both strongly positive log-odds scores, indicating that both scenarios are very likely in homologous RNA. However x_1, aligned to another 7-mer x_3 = **U**C-GU-**U**C-CG-**G**G-UA-**G**A (also with exactly four mismatched nucleotides), scores -4.28, indicating that such a match is unlikely in homologous RNA. For a positive threshold score such as $T = 30.0$, we say that x_1 and x_2 are *neighbours* of x_1, but x_3 is not. Choosing a threshold too low would result in finding all true hits but would also return a high number of false hits. While a higher threshold would decrease false positives, it may cause some true hits to be missed. In Section 3.7, we theoretically model true and false positives for a range of thresholds, and show how an appropriate value for T is chosen.

Next, for a threshold T, scoring matrix M and seed s of weight k, we compute neighbourhoods of pair kmers. Neighbourhoods allow us to differentiate "close" matches from

unrelated kmers and distant matches. Given the set P of pair symbols, the neighbourhood is computed for all $|P|^k$ kmers, where each kmer can have up to $|P|^k$ neighbours. We show in Section 3.6 that using $P = \{AU, CG, GC, UA, GU, UG\}$ is an acceptable tradeoff to using the entire set $P = \{AA, AC, AG, AU, CA, \ldots, UU\}$. Brejová *et al.* [3] have described an efficient algorithm to compute such neighbourhoods. Our simpler implementation for this step is briefly outlined in Section 4.2, which results in the Static Pair Neighbour (SPN) table being populated. The term "static" denotes that the table is independent of the target and query sequences.

The next step of hashing the target nucleotide sequence is crucial to, and forms the basis of, our pair seeding homology search method. The target is a *nucleotide* sequence in which locations of *pair kmers* have to be identified. Only then can alignments be made of query pair kmers to target ones. This steps resolves the nucleotide sequences in the target into high-scoring, potential pair kmers. To do so, we define the *window size* of hashing.

Definition 9 *The* window size w *of hashing is the maximum distance in nucleotides allowed between two nucleotides that pair to form a pair symbol.*

Note that the window size as defined above is independent of the query and target sequences. The only condition is that it be larger than the maximum arc of expected query pair kmers, as otherwise the algorithm will fail to find matches to the given query. This quantity is specified at the time the target is hashed, and we have used a default value of 300nt.

We are given a target sequence of length n, a query RNA sequence of length q, window size w, arc margin a and separation margin r. Along with a default window size of 300, we have used default values of 10 for the arc margin and 3 for the separation margin. For example, a 200 nucleotide query RNA that uses an arc margin of 10 allows for up to 10 nucleotides to be inserted in a homologous sequence. Our analysis of false positive rates in Section 3.7 uses the range 1–20 for the arc margin and values of 3 and 10 for the separation margin, ranges which are sufficient to identify homologs of the families we have experimented with.

To hash a target using straightforward nucleotide hashing, with a seed of weight k, takes $\Theta(nk)$ time. However, pair hashing involves additional work. First, each position may have to be hashed *twice* – once with the seed s, again with the reverse seed s^R – if

the seed is not palindromic. Then, for each nucleotide kmer so hashed, up to w candidate kmers have to be checked, in the worst case, to form potential high-scoring pair kmers homologous to the query. The worst-case runtime of the target hashing algorithm is thus $\Theta(2nk + nwk) = \Theta(nwk)$. We present our implementation in Section 4.3 that runs much faster on average. As a result of the target hashing step, locations of potential pair kmers in the target are stored in the Pair Hash table. Assuming the target is random noise DNA, the expected number N of pair kmers added to the Pair Hash table can be estimated as $N = nw(\frac{6}{16})^k(1 - f)$, where k is the seed weight and f is the fraction between 0 and 1 of the SPN table entries that have zero neighbours. We have experimentally estimated f (which is a function of T) over a range of 7-mer thresholds, and presented the values in Table 3.5. Thus, the number of Pair Hash table entries is $\Theta(nw)$.

Next, we process the query – a trivial operation that takes $\Theta(q)$ time – to identify $O(q)$ pair kmers in it. For the 1-hit method, checking each query kmer against possible candidates in the Pair Hash table takes $\Theta(nwq)$ time, since there are $O(q)$ query kmers and $\Theta(nw)$ Pair Hash table entries in the worst case. The expected number of seed hits is $\Theta(nwqa)$, from which it follows that false positives grow linearly with the arc margin.

For the 2-hit method requiring two pair kmer hits, the hashing time is still $\Theta(nwk)$. The hit finding time is $\Theta((nwq)^2)$ for a naïve implementation, in which $\Theta((nw)^2)$ pairs of target hits are examined for each of $\Theta(q^2)$ pairs of disjoint query kmers. We have improved this somewhat by partitioning the target into blocks of size b, so that the effective runtime is $\Theta(nb(wq)^2)$ – see Section 4.4.3. The number of seed hits is $\Theta((nwqa)^2 r)$, from which it follows that false positives for this method vary linearly with the separation margin and quadratically with the arc margin.

Finally, for the 2-hit method requiring one pair kmer hit and one nucleotide l-mer hit, the hash time is $\Theta(nwk + nl)$, since a pair hash table and a nucleotide hash table are required. The hit finding time is $\Theta(nb(wq)^2)$ for reasons analogous to the above case. The number of hits varies as $\Theta((nwq)^2 ar)$, from which it follows that false positives vary linearly with the arc margin and the separation margin. Note that the efficiency of the 2-hit algorithms we have presented can likely be improved considerably, for instance by using additional data structures to sort hits by arc, and further sorting pairs of hits by separation. We have not investigated these ideas, but identify them as potential future work in this area.

3.5 Pair Kmer Score Distributions

As mentioned earlier, Ribosum [12] matrices allow us to score the alignment of one nucleotide pair to another. Hence, these matrices can be used to score alignments of a pair kmer to another pair kmer. However, given a scoring matrix M, we still have to determine a suitable choice for the threshold score T, so that we can identify alignments that score above T as candidates to be extended using the exhaustive algorithm. In this section, we describe how an appropriate threshold score T is chosen, given a scoring matrix M. But before doing so, we briefly justify the use of the Ribosum-95 matrix in all our investigations.

The Ribosum matrices, as mentioned earlier, number 170 in all. The large number of matrices results from varying two parameters in the construction of these matrices. To choose a matrix for our experiments, we considered the alignment scores for perfect matches of the Watson-Crick and UG/GU pairs. These are presented for a range of matrices in Table 3.1 below.

Table 3.1: Ribosum matrix main diagonal scores. For example, the score for a GC-to-GC alignment in the Ribosum-85 is 5.51. Please see the text for a discussion of these values.

Matrix	AU	CG	GC	GU	UA	UG
Ribosum100	4.25	5.22	5.49	3.32	4.72	3.19
Ribosum95	4.22	5.22	5.5	3.31	4.71	3.18
Ribosum90	4.18	5.21	5.5	3.29	4.69	3.16
Ribosum85	4.09	5.2	5.51	3.23	4.6	3.11
Ribosum80	3.95	5.2	5.52	3.11	4.42	2.97
Ribosum75	3.79	5.29	5.63	2.96	4.15	2.8
Ribosum70	3.77	5.3	5.65	2.97	4.11	2.76
Ribosum65	3.57	5.44	5.8	2.9	3.97	2.59
Ribosum60	3.34	5.64	6.02	2.76	3.85	2.55
Ribosum55	3.66	5.52	5.84	3.06	4.19	2.9

Table 3.1 shows that the scores change abruptly in the Ribosum-60 matrix, relative to the matrices before and after it. Further, scores in the lower half of the rows are more variable from row to row, as compared to those in the Ribosum-90, -95 and -100 matrices. The *stability* of values is a factor in favour of choosing a matrix from the upper rows.

Secondly, we computed the "GC-to-AU" ratio, or the ratio of GC-GC to AU-AU

scores. This ratio is greater in the lower rows of the table – for the Ribosum-60 matrix, it is 6.02/3.34, while for the Ribosum-95 it is 5.5/4.22. A large GC-to-AU ratio is undesirable, since such a matrix would score poorly even perfect alignments of AU-rich paired regions, at the expense of GC-rich ones. In contrast, a matrix such as the Ribosum-95 gives relatively closer scores to matches of AU-rich and GC-rich regions. Ribosum matrices are derived from a specific RNA family (SSU rRNA) and imply pair frequencies inherent in that family. But it is possible that a homology search involves a query RNA sequence with very different pair frequencies. Hence, to prevent skewing our algorithm strongly against AU-rich families, we have chosen the Ribosum-95 matrix – from the mid-range of matrices with low, stable GC-to-AU ratios – for our experiments.

The Ribosum scores are log-base-2 odds ratios, and hence from the matrix, we can derive the underlying probabilities of aligning each pair to every other pair, to generate a probability matrix that is analogous to the score matrix in Table 2.2. This probability matrix is shown in Table 3.2.

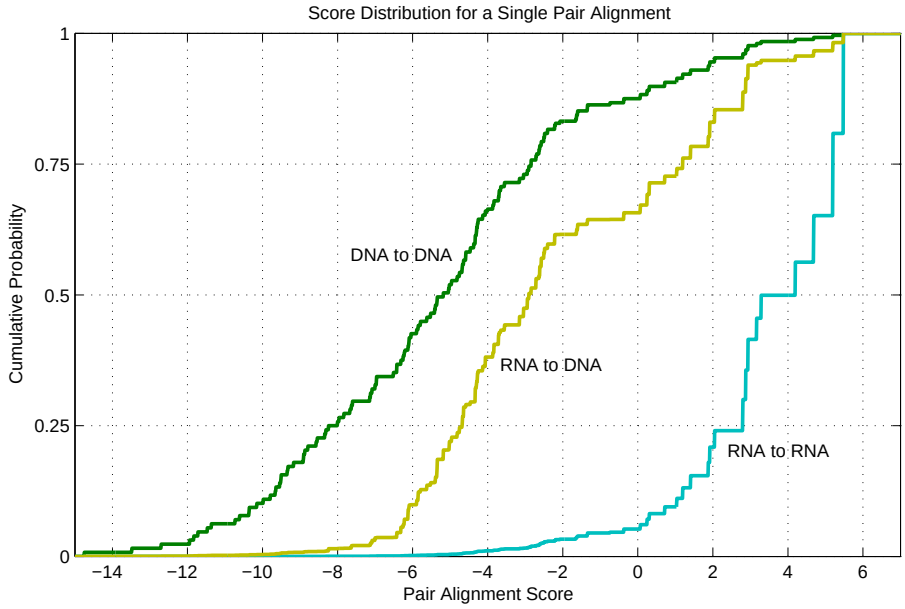

Figure 3.7: Score distribution for a single pair alignment. The curves from left to right are distributions for: (1) random DNA pair aligned to random DNA pair, (2) RNA pair aligned to random DNA pair, (3) related RNA pairs aligned. The median score for true alignment of pairs is 3.8, and for false alignments is -2.9.

Table 3.2: Ribosum-95 pair to pair alignment probability matrix [12]. Each cell is the probability ($\times 10^{-5}$) of finding the corresponding pairs aligned in related RNA sequence. Note that the matrix is generated such that the triangle values sum to 1.0. Hence, to convert this into rectangular probability matrix, every non-diagonal entry [a, b] is halved and replicated into entry [b, a].

	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
AA	63															
AC	2	77														
AG	1.8	1.3	230													
AU	25	210	15	6300												
CA	0.76	0.49	0.67	13	4.4											
CC	0.057	1.2	0.027	100	0.2	17										
CG	61	18	160	3100	84	19	15000									
CU	0.16	0.35	1	32	1.7	3.8	31	37								
GA	8.4	4.3	1.4	61	3.4	0.11	17	3.8	180							
GC	24	110	14	5600	18	47	5900	130	160	19000						
GG	1.4	0.88	65	36	0.35	0.25	45	1.3	1.8	45	170					
GU	10	29	7.5	1200	11	5.0	1000	29	13	2300	25	4300				
UA	130	15	120	2500	88	7	5800	28	33	2920	13	820	8900			
UC	0.18	1.1	3.0	29	2.4	1.2	33	20	5.3	38	0.51	27	49	15		
UG	71	5.4	18	590	11	2.1	2000	7.7	5.1	1000	60	350	1600	34	4000	
UU	1.4	1.8	1.0	110	1.2	5.2	79	11	0.87	85	27	100	190	7.6	52	150

Given the scoring matrix and corresponding probability matrix for alignment of a *single* pair to a pair, we can trivially compute the cumulative probability distribution (against the alignment score) for a single pair, for example, an AU-to-GC alignment. The results are shown in Figure 3.7, in the form of three curves:

1. The right-most curve is of the score of an RNA pair aligned to another RNA pair. We used the probability matrix generated by RSEARCH [12] shown in Table 3.2, along with corresponding scores in Table 2.2. The distribution favours strongly positive scores, as expected. About 95% of the time, such an alignment gives a positive score. The rest of cases are due to GU-to-UG matches (which score negatively) and alignments of pairs not composed of complementary bases.

2. The leftmost curve is the score distribution of aligning a random DNA pair to another random DNA pair, but scoring with the Ribosum matrix. For this, we generated a probability matrix (such as the one in Table 3.2) to capture probabilities of pairs in random DNA. The details of the calculations are in Section 4.1.

3. The middle curve is the distribution of scores for a random DNA pair aligned to an RNA pair. In this case, we generated a matrix that captured the probabilities of aligning one RNA pair to a pair in random DNA. (Please see Section 4.1 for the details.) In this case, the expected scores are still negative, though less so than the first curve. This is so since in DNA all 16 pairs are more or less equally likely. But in RNA the 6 pairs that are very frequent are high-scoring, the other 10 being rare, so the RNA pair biases us towards more high-scoring pairings.

From the distribution in Figure 3.7, we can derive the alignment score distribution for a pair *kmer*, that is, a sequence of k pairs aligned to another sequence of k pairs, where the pair in each position has score distributed as per Figure 3.7. An algorithm to do this based on polynomial convolution has been described by Brejová *et al.* [3]. We present our implementation in Section 4.1. The resulting score distributions, for pair 4-mers and 7-mers, are given in Figures 3.8 and 3.9 respectively.

These distributions allow us to choose a threshold score T.

Let us define the "hit rate" of a threshold score to be the probability that a pair kmer within an alignment will score *above* that threshold. Hence, if we choose a threshold score so that 98% of the scores lie below it, then that threshold has a hit rate of 2%. We can

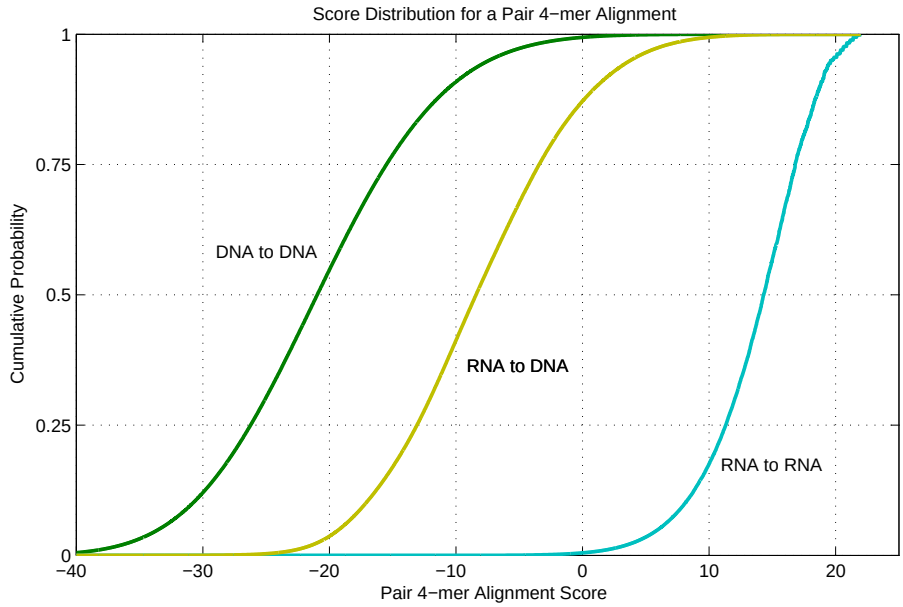

Figure 3.8: Score distribution for pair 4-mer alignment. The curves from left to right are distributions for: (1) random DNA aligned to random DNA, (2) RNA pairs aligned to random DNA, (3) related RNA pairs aligned. The median score for true hits is 14.31, and -8.41 for false hits (RNA to DNA).

relate the hit rate r to the approximate number of trials that are required to achieve a seed hit. Assuming independent, identically distributed trials, if we choose integer n to satisfy $(1-r)^n \leq 0.2$, then we will achieve a seed hit 80% of the time, by using n trials, that is, n independent query kmers. (While the independence assumption is invalidated by overlapping kmers, it is sufficient for the purposes of estimation here.)

From the data in Figures 3.9 and 3.8, we can identify the threshold score that relates to any given hit rate, for a particular seed size. Table 3.3 shows a range of hit rates and their corresponding thresholds – for pair 4-mers, 6-mers and 7-mers.

Table 3.3 also gives the approximate number of kmers required for a hit 80% of the time, for that particular hit rate of a single trial. Note that the BLAST seed of size 11 has a 2% hit rate at a homology level of 0.7. Hence on average, 80 11-mers are required to achieve a hit 80% of the time, since $1-(1-0.2)^{80} \approx 0.8$.

Once a threshold is chosen (*e.g.* $T = 31.49$) for a given seed size (*e.g.* $k = 7$) we can also compute the expected false positives for those choices. This can be done from the

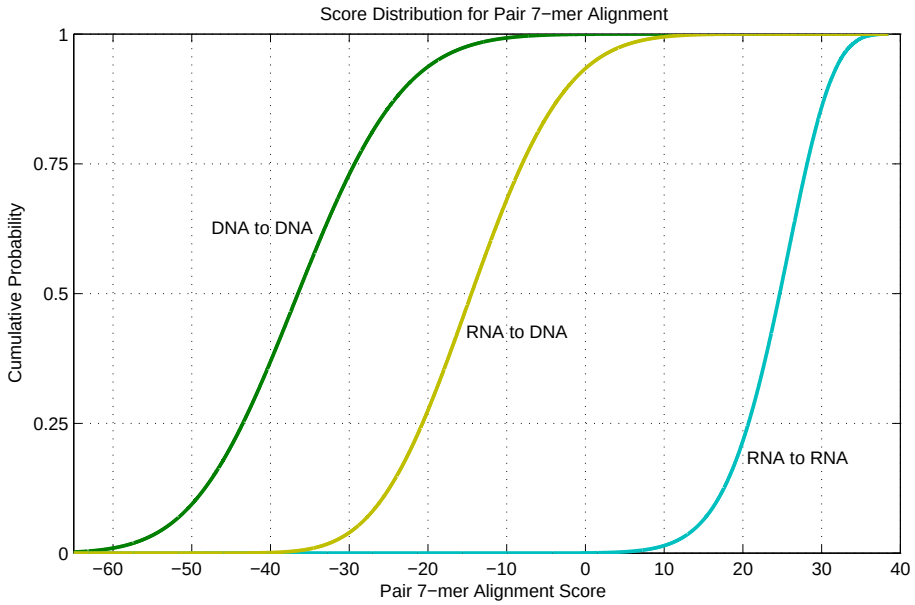

Figure 3.9: Score distribution for pair 7-mer alignment. The curves from left to right are distributions for: (1) random DNA aligned to random DNA, (2) RNA pairs aligned to random DNA, (3) related RNA pairs aligned. The median score for true hits is 24.69, and -14.47 for false hits (RNA to DNA).

Table 3.3: Hit rates and threshold scores for 4-mers, 6-mers and 7-mers using the Ribosum-95 matrix. For example, to achieve a hit rate of 5.2% using 6-mers, a threshold score of 28.17 or lower has to be used. Using $T = 28.17$, we expect on average that 30 query kmers would be required to give at least one seed hit 80% of the time.

Cumulative Probability (CP)	Hit Rate (= 1 - CP)	# Kmers Needed	Scores		
			4-mer	6-mer	7-mer
98.000%	2.0%	80	20.87	29.55	33.90
97.353%	2.6%	60	20.60	29.22	33.46
96.056%	3.9%	40	20.11	28.67	32.79
94.777%	5.2%	30	19.60	28.17	32.28
92.269%	7.7%	20	19.10	27.43	31.49
89.826%	10.2%	15	18.71	26.87	30.85
85.133%	14.9%	10	18.08	25.98	29.86

curves in Figures 3.8 and 3.9. To calculate the rate of false positives, the middle curve (and not the leftmost one) would be considered. This is because query kmers, being from known RNA molecules, are distributed as per the RNA pair frequencies, while the target is random DNA. The curves immediately show, as expected, that a larger seed size gives a greater separation (and hence, better ability to distinguish) between true and false hit curves.

However, an alternate method to compute false positives is to use directly the values of average number of neighbours for a given threshold. We choose the latter approach and present the expected false positive values in Section 3.7.

3.6 Composition of Pair Kmers

The above score distributions and the concept of neighbourhood consider 16 possible pair symbols in each position in a pair kmer. However, as can be seen from the scoring matrix in Table 2.2, only 20 of the total 136 scores are positive. In particular, 6 pairs are very frequent in RNA – the *relevant* pairs – and others are very rare. From the Ribosum-95 probabilities, it can be deduced that the probability of a relevant pair is 0.975, while that of a non-relevant pair is 0.025. For the purposes of this discussion, kmers composed solely of relevant pairs are called *relevant kmers*. For example, AU-GC-UG-GU is a relevant 4-mer; AU-GG-CA-UA is not since it contains the pairs GG, CA.

Since the 10 non-relevant pairs are rare in RNA, can we ignore them in hashing pair kmers? Doing this results in smaller memory requirements for hash tables. For example, there are $16^7 = 268,435,456$ entries in the space of all 7-mers of 16 pair symbols, and $6^7 = 279,936$ entries in the space of 7-mers of relevant pairs only.[1]

The latter size is aprroximately 1000 times smaller than the former, so we would have a significantly smaller hash table by using only relevant pairs. Despite this benefit, using relevant kmers is potentially less accurate. For example, the 7-mers x = GC-GC-GC-GC-GC-GC-GC and y = GC-GC-GC-GC-GC-GC-AC score 30.3 when aligned to each other, though the AC pairing is non-standard. For a threshold $T = 30.0$, they would properly be considered neighbours. But by hashing relevant 7-mers only, we would miss this since

[1] In practice, efficient hashing requires a *fixed* number of bits per symbol, so the total number of hash entries is greater than these values. Still, 16 symbols require four bits per symbol while a 6-symbol alphabet only requires three bits per symbol – a saving of 7 bits if hashing pair 7-mers.

y is not a relevant 7-mer. We seek here to quantify how often such neighbours, that are not relevant kmers, occur. In particular, given a particular relevant pair kmer p and a threshold T, how many neighbouring kmers does p have in the space of *all* kmers and in the space of *relevant* kmers? If the difference in these two values is small on average – over the space of all relevant kmers, for the thresholds generally used – we could conclude that hashing with relevant kmers only is an acceptable approximation.

Table 3.3 shows some thresholds and their corresponding hit rates. Note that using a threshold with a hit rate higher than, say, 14% will give more hits, but also result in a large number of false positives – see Section 3.7. Hence such thresholds are not of practical use. We have chosen the four lowest score thresholds from Table 3.3 – these would give the largest difference between the number of relevant neighbours and all neighbours. For a given seed size and threshold (*e.g.* 7-mers with $T = 30.85$), we computed two values: RN, the number of relevant neighbours, and AN, the number of all neighbours, for every relevant kmer – using the algorithm in Section 4.2.1. For each kmer, we found the difference between the two values computed, as a fraction of AN. This process was performed for 4-mers, 6-mers and 7-mers, for four thresholds in each case, for a total of twelve cases.

Table 3.4: Difference in the number of *all* and *relevant* neighbours for 4-mers, 6-mers and 7-mers. Columns 2, 3, 4 give the average difference between the number of *all* and *relevant* neighbours for the threshold and kmer size specified, as a fraction of all neighbours. For example, the entry in row 3, column 3 (1.66E-6) says that for the average relevant pair 7-mer at $T = 30.85$, the number of total neighbours exceeds the number of relevant kmers by a factor of 1.00000166.

Hit Rate	Average Difference in No. of Neighbours		
	for 4-mers	for 6-mers	for 7-mers
5.223%	0	0	0
7.731%	0	0	0
10.174%	0	0	1.66×10^{-6}
14.866%	0	0	3.65×10^{-5}

The results for the twelve cases are summarized in Table 3.6. The values demonstrate that, on average, the number of all and relevant neighbours are very close to each other, differing by factors of $(1 + \epsilon)$, ϵ being the fractions in Table 3.6. For smaller seed sizes, that is, for 4-mers (which are used in a 2-hit approach) and 6-mers, there is no difference

in accuracy between using relevant and all kmers.

While the average differences are small, we also looked at the *overall distribution* of the difference between AN and RN values for the two lowest thresholds for pair 7-mers: T = 29.86 and 30.85. Both of these are among the lowest practically used thresholds and represent worst-case behavior.

For T = 29.86, we found that only 2,024 7-mers of a total of 279,937 have non-zero differences between AN and RN values – about 0.7% of the total space of 7-mers. The average difference over all 7-mers was 3.65×10^{-5}. For T = 30.85, only 98 of 279,937 7-mers have non-zero difference values – only 0.03% of the space of 7-mers. In this case, the average difference is 1.66×10^{-6}.

For both thresholds above, for a small fraction of 7-mers over the entire range, the difference value peaks well above the average. For example, the average difference for the T = 29.86 case is 3.66×10^{-5}, but the peak value is 3.66×10^{-2}, about 1,000 times larger. For T = 30.85, the peak value is 5.59×10^{-3}, compared to the average of 1.66×10^{-6} – a difference of a factor of 3,367. While these are large deviations from the average, they are still small in absolute terms: at its peak the difference between AN and RN values is the factor 1.0366 for $T = 29.86$ and 1.00559 for $T = 30.85$. More importantly, note that for higher thresholds that would be actually used in practice (such as $T = 31.49$), the factors are zero. Hence, we conclude that hashing using relevant pairs is an acceptable tradeoff.

Using relevant pairs only, we have computed distributions of number of pair kmer neighbours for several pair 7-mer thresholds, capturing the *average* as well as the *maximum* numbers in each case. The results are given in Table 3.5.

3.7 Theoretical Sensitivity and Specificity

It is clear from the score distributions shown earlier that a large seed size results in lower false positives for the same true positives. For example, the threshold that achieves a 2% hit rate for pair 7-mers (T = 33.90) gives a lower false positive rate than the threshold (T = 29.55) that achieves the same hit rate for pair 6-mers. This means that a larger seed would result in better performance of the search algorithm.

Two factors prevent the use of an arbitrarily large seed. Firstly, a larger seed would require a larger Static Pair Neighbour (SPN) table. For example, pair 11-mers would

Table 3.5: Number of neighbours *versus* threshold score for pair 7-mers. For example, with $T = 31.49$, there are on average 2.53 pair 7-mers that align with any given pair 7-mer to score above 31.49. Further, there is a pair 7-mer that aligns with 288 pair 7-mers to score above 31.49, and no other pair 7-mer has more than 288 neighbours. Finally, using $T = 31.49$ results in a hash table that has 66.0% of the entries empty, that is, with zero neighbouring 7-mers.

Threshold Score	# Neighbouring 7-mers		% Empty Entries
	Average	Maximum	
33.90	0.146	29	92.8%
33.46	0.26	29	89.5%
32.79	0.578	120	83.3%
32.28	1.05	162	77.2%
31.49	2.53	288	66.0%
30.85	4.91	351	54.8%
29.86	13	742	39.2%

require a table with 6^{11} or 362 million entries. If each pair 11-mer had only 20 neighbours on average, the table would require about 29 Gbytes of memory. Since this table is needed during target hashing and hit finding, it has to be maintained in memory during those steps. This would tax the resources of a typical machine and limit the maximum pair seed size.[2]

In the case of RNA molecules, a second factor limits how large the seed can be. There is a fair amount of variability in the structures of RNA molecules. Some of them have very little paired content, and consist largely of unpaired sequence with occasional pairs. An example, the small nucleolar RNA U29, is shown in Figure 3.10. Other RNA molecules like the Plasmid RNAIII, shown in Figure 3.11, are distinguished by large, contiguous regions of pairing. Still other RNAs contain paired content, but in a fragmented fashion, such as the CsrB RNA (carbon storage regulator) shown in Figure 3.12. In practice, it is rare to find paired regions in typical RNA sequences that are longer than 8 pairs. Even if one member of a family has a paired region of length 10, it is common that a related RNA has inserted nucleotide(s) within that region, breaking it into two or more smaller

[2]Note that the Pair Hash table does not grow in the same manner with increasing seed size. For example, pair-hashing a random 250,000 nucleotide target sequence using pair 7-mers would require about 200 Kbytes of memory, in addition to the memory footprint of the empty Pair Hash table. However, pair-hashing the same target using pair 11-mers would require at most an additional 8 Kbytes! These estimates are based on the expressions in Section 3.4.

paired regions. For example, suppose we hash a query RNA sequence using pair 10-mers, and search for hits to one 10-mer q. If a true homolog of q in the target sequence has mutated to become two paired 5-mers, hashing using 10-mers would miss it. Hence, for practicality, we limit the length of pair seeds in this study to 7-mers or less. Further, we use continuous seeds, that is seeds with the same weight and length, in our work, due to the restrictive lengths of paired regions.

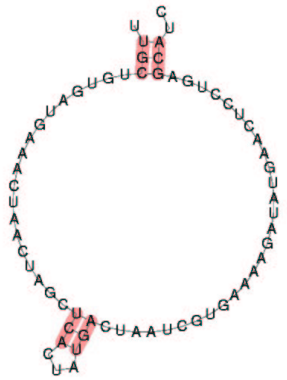

Figure 3.10: Small nucleolar RNA U29 structure [9]

We have already related threshold scores to true hits rate, for various seed sizes, as shown in Table 3.3. Here, we relate the same thresholds to false positive rates, for 1-hit and 2-hit methods outlined earlier.

Table 3.5 shows how the average and maximum number of neighbouring pair 7-mers varies with the threshold score. To compute the false positive rates in this section, we use the *average* number of neighbouring pair kmers, as shown in this table. For example, using T = 31.49, we know that there are 2.53 pair 7-mers (out of a maximum of 16^7 in random DNA) that align with any given 7-mer to score above the threshold. Hence the probability of a false hit at a fixed target position is $2.53/16^7$. We have also shown the maximum number of neighbours in Table 3.5 to highlight that, performance can be noticeably poorer in the worst case than in the average case. For example, $T = 31.49$ has an average of 2.53 and a maximum of 288 neighbours – a difference of two orders of magnitude. This means that to get predictably better performance from a pair seed, we would ideally like to have about two orders of magnitude difference between the false hit rate of the BLAST seed and that of our pair seed with an appropriately chosen threshold.

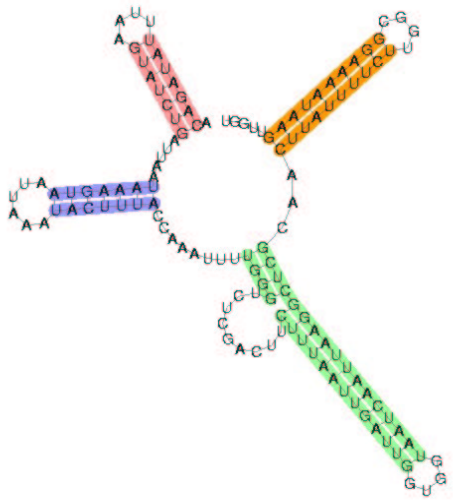

Figure 3.11: Plasmid RNAIII structure [9]

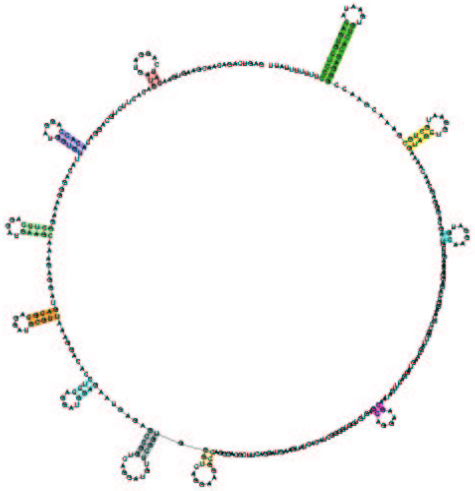

Figure 3.12: CsrB (carbon storage regulator) RNA structure [9]

Note that this situation is not specific to RNA homology searches, since it could occur in the case of DNA searches as well. Consider a 60% A-T and 40% G-C sequence. The probability of a match at one position on average is $2 \times 0.3^2 + 2 \times 0.2^2 = 0.26$. So an 11-mer on average matches with probability 0.26^{11}, while the 11-mer AAAAAAAAAAA matches with probability 0.3^{11}, which is 5 times the average, and 90 times the probability of the 11-mer GGGGGGGGGGG matching, which is 0.2^{11}.

3.7.1 False Positives for 1-hit Methods: Pair 7-mers and 6-mers

The simplest homology search using pair seeds is to use a 1-hit approach that involves finding a hit to a given pair kmer.

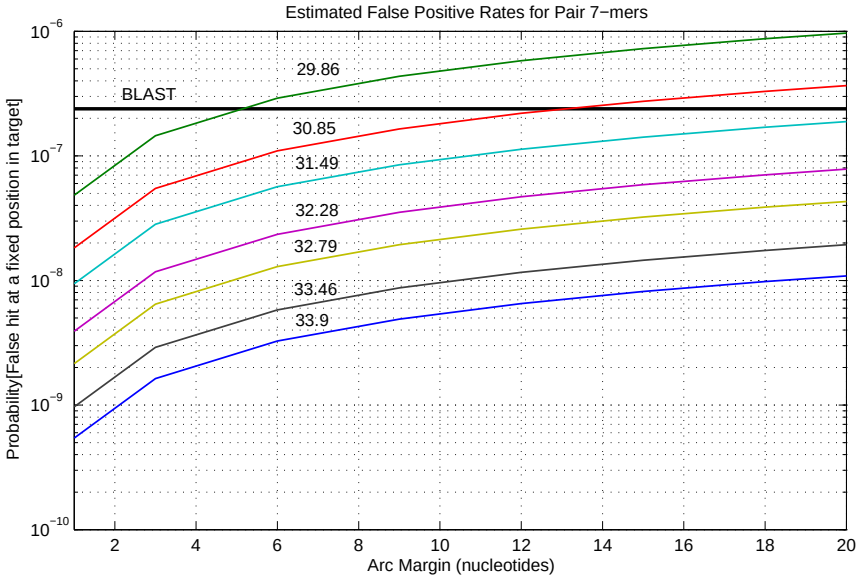

Figure 3.13: Estimated false positive rates for pair 7-mers. Each curve gives the theoretical probability of a false positive hit for the given threshold score, over a range of values for the arc margin. The dark horizontal line is the false positive rate for the BLAST size-11 seed, assuming equal nucleotide probabilities. Over a range of thresholds and arc margins, pair 7-mers outperform the BLAST seed.

In this case, we estimate the false positive rate for arc margin a as Pr[false hit of pair k-mer] $= 1 - (1 - p)^a$, where $p =$ Pr[false hit of pair k-mer with $a =$ 1nt]. For the sake of comparison, the figures also show where the false hit probability of the BLAST size-11

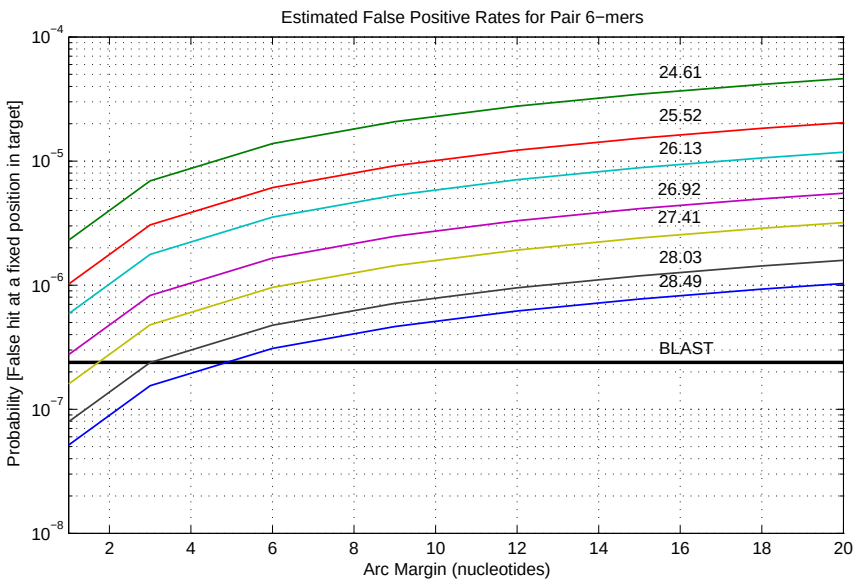

Figure 3.14: Estimated false positive rates for pair 6-mers. Each curve gives the theoretical probability of a false positive hit for the given threshold score, over a range of arc margin values. The dark horizontal line is the false positive rate for the BLAST size-11 seed, assuming equal nucleotide probabilities. The BLAST seed has lower false positives compared to, and hence performs better than, pair 6-mers over a wide range of thresholds and arc margins.

seed falls. Figures 3.14 and 3.13 show the estimated probability of a single false positive hit (at a *fixed* location in the target sequence) for pair 6-mers and 7-mers, respectively. From Figure 3.14 it is evident that for most thresholds, the false positive rates for pair 6-mers are higher than those achieved by the BLAST seed. This indicates that pair 6-mers would give poor performance in practice, compared to a simple BLAST-based search. Pair 7-mers fare better, with expected false positive rates for several thresholds (*e.g.* $T = 31.49$) being lower than the BLAST seed's, if the arc margin is low enough. Since the arc margin used would most often be in the 5 – 10 nucleotide range, a 1-hit approach using pair 7-mers is possibly a better alternative to a BLAST heuristic. We investigate the performance of 1-hit pair 7-mers and present the results later in this work. Below, Table 3.6 relates the hit rate (that is, the true positives rate) to the false positive rate of pair 7-mers.

Table 3.6: True *versus* false positives for pair 7-mers. We relate the true hit rate of each threshold to the false hit rate *fp* shown below, based on an arc margin of 1 nucleotide. For a larger arc margin *am*, the false positive rate is $1-(1-fp)^{fp}$. For comparison, the BLAST seed achieves a true hit rate of 0.020 and a false hit rate of 2.38×10^{-7}, assuming random sequence with equal base probabilities.

Threshold Score	True Hit Rate	False Hit Rate
33.90	0.020	5.44×10^{-10}
33.46	0.026	9.69×10^{-10}
32.79	0.039	2.15×10^{-9}
32.28	0.052	3.91×10^{-9}
31.49	0.073	9.42×10^{-9}
30.85	0.101	1.83×10^{-8}
29.86	0.149	4.84×10^{-8}

3.7.2 False Positives for 2-hit Method: Two Pair 4-mers

As was mentioned earlier, it is not uncommon to see paired regions in RNA molecules be interrupted by single unpaired nucleotides. Hence, we investigated false positive rates for a 2-hit method that requires two *non-overlapping* or disjoint pair 4-mer hits. Both 4-mer hits use the same threshold score. Since two hits are chained together to give an extensible hit, this method uses an additional parameter called the *separation margin*, discussed in Section 3.3. This quantity is analogous to the "distance" between the two hits. The margin determines how much variability is tolerated between the distances in the query and target sequences.

Figures 3.15 and 3.16 show expected false positive rates for separation margins of 3 and 10 nucleotides, respectively. The false positive rates vary linearly with the separation margin.[3] We compute the probability of a false hit here as the product of the probability of a false 4-mer hit, and that of a having at least one 4-mer hit within a window of size equal to the separation margin *s*. Thus, if p is the probability of a false hit of one pair 4-mer for a given threshold and arc margin, then the probability of a false hit of two pair 4-mers is approximately $p \times [1-(1-p)^s]$. This computation assumes independence of overlapping hits, which is not strictly accurate, but is sufficient for the purposes of

[3]Note that the rates vary *quadratically* with the arc margin, since there are two hits, each of which has a false hit rate that varies linearly with its arc margin; if we double the frequency of false pair hits, the frequency of false 2-hit matches is quadrupled.

estimation.

The resulting plots indicate that the false positive rates for frequently used thresholds (*e.g.* $T = 19.6$ for 4-mers) are approximately as good as those of 1-hit pair 7-mers. This reinforces our idea that the 2-hit method in general could work as well as the 1-hit pair 7-mer approach, with the added advantage that fragmented paired regions (of length 4 or more) could be hashed using this method.

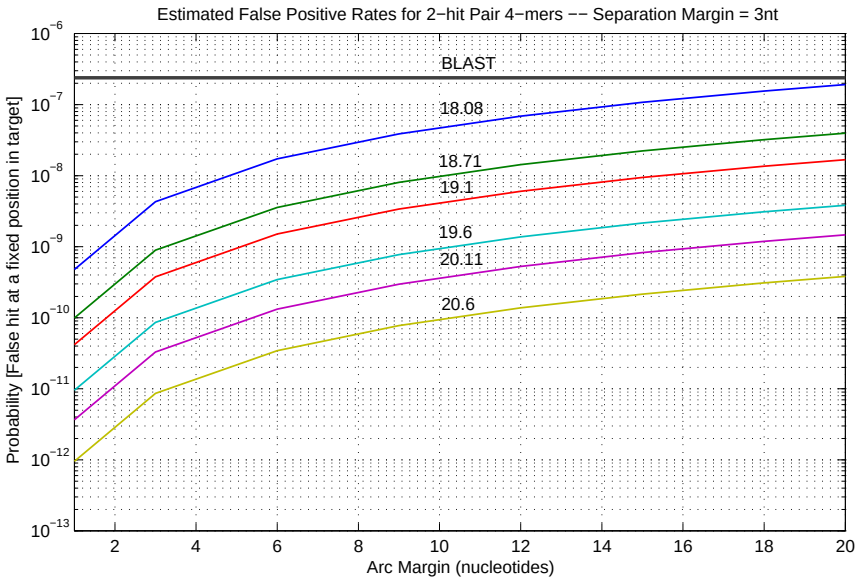

Figure 3.15: Estimated false positive rates for 2-hit pair 4-mers, separation margin = 3 nucleotides. Each curve gives the theoretical probability of a false positive hit made of two 4-mer hits, for the given threshold 4-mer score. The x-axis shows the arc margin value used for each single 4-mer hit. The separation margin is the variation allowed in the separation of the two hits between the query and target. The dark horizontal line is the false positive rate for the BLAST size-11 seed, assuming equal nucleotide probabilities.

3.7.3 False Positives for 2-hit Method: Pair 4-mer + Nucleotide 8-mer

Finally, we present estimated false positive rates for the 2-hit method that requires one hit to be a pair 4-mer, but the second hit to be a *nucleotide* 8-mer hit. As was shown earlier, some RNA families have low paired content. For such queries, a 2-hit method that only hashes pair kmers would ignore the majority of the content (that is, unpaired regions) in the RNA query. This method, in contrast, takes into consideration unpaired

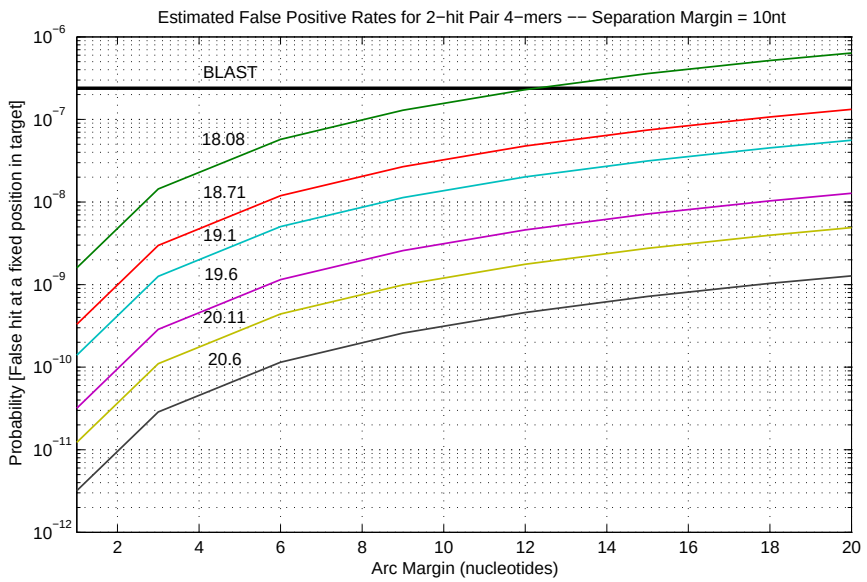

Figure 3.16: Estimated false positive rates for 2-hit pair 4-mers, separation margin = 10 nucleotides. Each curve gives the theoretical probability of a false positive hit made of two 4-mer hits, for the given threshold 4-mer score. The x-axis shows the arc margin value used for each single 4-mer hit. The separation margin is the variation allowed in the separation of the two hits between the query and target. The dark horizontal line is the false positive rate for the BLAST size-11 seed, assuming equal nucleotide probabilities. A higher separation margin results in higher false positives, for the same thresholds.

content in RNA molecules in performing the search, and hence improves the sensitivity of the search. We have shown here the computations for a single case: one nucleotide 8-mer chained to a pair 4-mer. As in the case above, the separation margin is set to 3 and 10 nucleotides, to generate the plots shown in Figures 3.17 and 3.18.

Figures 3.17 and 3.18 show that, for the given ranges of thresholds and arc margin values, the false positives for this approach are noticeably superior to those of the earlier two approaches. We investigate the practical performance of all three approaches later in this work.

In closing, we note that keeping false positives low ensures that the hit extension phase, that follows the hit finding phase, is executed on fewer potential matches, decreasing the overall runtime. However, the overall runtime of a homology search method depends on more than the false positive rate for 2-hit models. Though the 2-hit models may offer

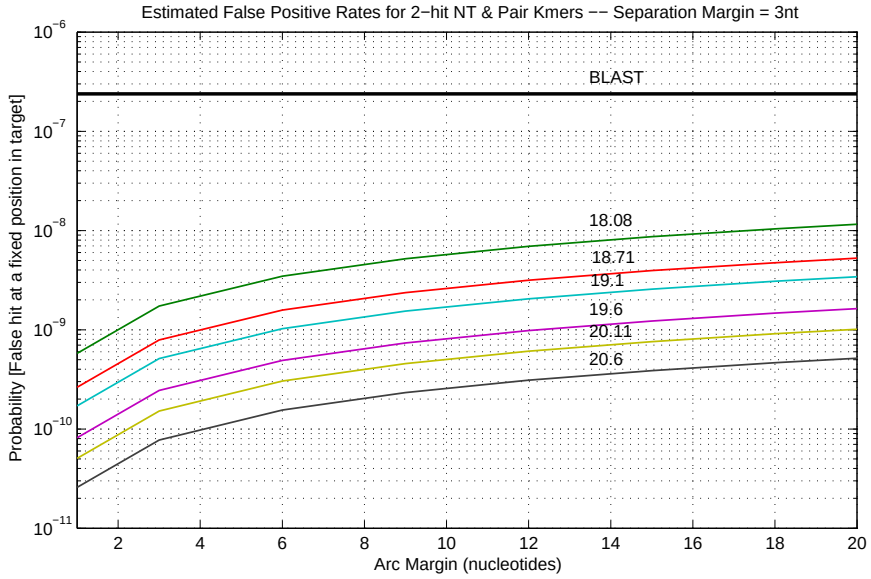

Figure 3.17: Estimated false positive rates for 2-hit made of 1 nucleotide 8-mer + 1 pair 4-mers with separation margin = 3 nucleotides. Each curve gives the theoretical probability of a false positive hit made of two smaller hits: a nucleotide and a pair hit, for the given threshold 4-mer score. The x-axis shows the arc margin value used for the pair 4-mer hit. The separation margin is the variation allowed in the separation of the two hits between the query and target. The dark horizontal line is the false positive rate for the BLAST size-11 seed, assuming equal nucleotide probabilities.

lower false positive rates, the runtime overhead they incur in the hit finding phase is considerable, in comparison to the 1-hit model, as we note in Section 5.4.1. Further, the choice of the hit extension algorithm, which we do not consider in this work, would also have a big impact on the overall runtime.

3.8 Neighbourhood and Partnership Distributions

In this section, we provide details of the distributions of neighbourhoods for pair 7-mers. In addition, we introduce the concept of nucleotide kmer partnerships and present details of their distributions as well.

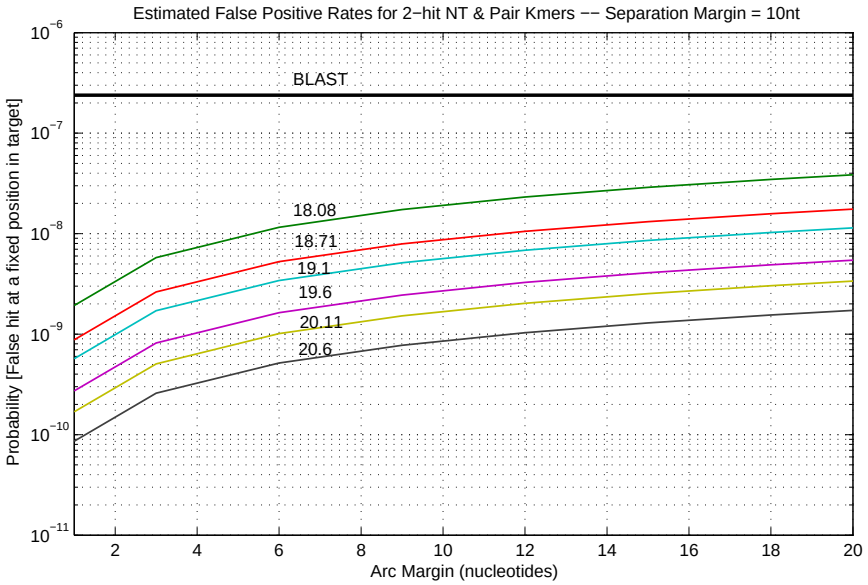

Figure 3.18: Estimated false positive rates for 2-hit made of 1 nucleotide 8-mer + 1 pair 4-mers with separation margin = 10 nucleotides. Each curve gives the theoretical probability of a false positive hit made of two smaller hits: a nucleotide and a pair hit, for the given threshold 4-mer score. The x-axis shows the arc margin value used for the pair 4-mer hit. The separation margin is the variation allowed in the separation of the two hits between the query and target. The dark horizontal line is the false positive rate for the BLAST size-11 seed, assuming equal nucleotide probabilities.

3.8.1 Number of Pair Kmer Neighbours

As noted previously, varying the threshold score for a pair kmer hit changes the contents of static neighbourhood tables. To better understand these changes, we chose four threshold scores for pair 7-mers, and computed the distribution of the number of neighbours for all kmers, for each choice of threshold. The results are summarized in Tables 3.7 and 3.8 below. These tables show that the majority of the 7-mers have relatively low number of neighbours, in the lower 25%-33% of the range. For example, with T = 31.49, the number of 7-mers with more than 90 neighbours is less than 0.2% of the total number of 7-mers.

Table 3.7: Distribution of neighbours for T = 29.86 (left) and 31.49 (right). For example, for T = 29.86, there are 268,637 7-mers that have between 0 and 74 neighbours. The average number of neighbours for each threshold is 13.0 and 2.53, respectively.

No. of Neighbours	No. of Kmers
0 – 74	268637
75 – 149	6321
150 – 224	2597
225 – 299	1393
300 – 374	393
375 – 449	427
450 – 524	70
525 – 599	35
600 – 674	35
675 – 749	29

No. of Neighbours	No. of Kmers
0 – 29	275140
30 – 59	3206
60 – 89	889
90 – 119	210
120 – 149	316
150 – 179	77
180 – 209	70
210 – 239	21
240 – 269	7
270 – 299	1

Table 3.8: Distribution of neighbours for T = 32.79 (left) and 33.9 (right). The average number of neighbours for each threshold is 0.578 and 0.146, respectively.

No. of Neighbours	No. of Kmers
0 – 12	275833
13 – 25	3368
26 – 38	574
39 – 51	98
52 – 64	35
65 – 77	21
78 – 90	0
91 – 103	0
104 – 116	0
117 – 129	8

No. of Neighbours	No. of Kmers
0 – 2	277128
3 – 5	1610
6 – 8	210
9 – 11	140
12 – 14	252
15 – 17	365
18 – 20	161
21 – 23	42
24 – 26	21
27 – 29	8

3.8.2 Number of Nucleotide Kmer Partners

Once we have derived the neighbourhood relations between pair kmers, we can then infer from that distribution the one for partnerships of nucleotide kmers of the same size. Briefly, if a pair 7-mer GC-GC-GU-GC-AU-AU-AU has at least one neighbour for a given threshold, then the nucleotide 7-mer GGGGAAA is a "partner" of UUUCUCC. Essentially, two nucleotide kmers are partners if and only if they form a complementary paired region between them. Section 4.2.2 outlines how the table with these partnerships (the Static Nucleotide Partner table) is built. Tables 3.9 and 3.10 summarize the results.

Table 3.9: Distribution of nucleotide partners for T = 29.86 (left) and 31.49 (right). For example, for T = 31.49, there are 6,720 nucleotide 7-mers that have between 3 and 5 partners. The average number of partners for each threshold is 10.39 and 5.81, respectively. The fraction of kmers with no partners is 0.01% and 0.99%, respectively.

No. of Partners	No. of Kmers
0 – 6	4239
7 – 13	8232
14 – 20	2499
21 – 27	1148
28 – 34	245
35 – 41	0
42 – 48	21
49 – 55	0
56 – 62	0
63 – 69	1

No. of Partners	No. of Kmers
0 – 2	2279
3 – 5	6720
6 – 8	4256
9 – 11	2310
12 – 14	469
15 – 17	273
18 – 20	21
21 – 23	56
24 – 26	0
27 – 29	1

Table 3.10: Distribution of nucleotide partners for T = 32.79 (left) and 33.9 (right). For example, for T = 32.79, there are 2,499 nucleotide 7-mers that have between 14 and 20 partners. The average number of partners for each threshold is 2.85 and 1.23, respectively. The fraction of kmers with no partners is 10% and 36%, respectively.

No. of Partners	No. of Kmers
0 – 1	5142
2 – 3	5999
4 – 5	3409
6 – 7	1484
8 – 9	148
10 – 11	140
12 – 13	42
14 – 15	0
16 – 17	21

No. of Partners	No. of Kmers
0	5939
1	5090
2	2639
3	1701
4	553
5	280
6	168
7	14
8	1

Chapter 4

Algorithms and Implementation

In this section, the different algorithms we developed to implement our ideas are presented and analyzed.

4.1 Building Kmer Score Distributions

Here, we describe how we generated the score distributions for pair 4-mers, 6-mers and 7-mers, given the scoring matrix for aligning pairs to pairs. As given in Section 3.5, these distributions were used to determine appropriate choices of the threshold score T.

We downloaded the RSEARCH package [12], including its source code, along with the complete multiple sequence alignment of the SSU rRNA family. With its source code, we extracted frequency counts and probabilities of alignment for each set of pairs.

4.1.1 Alignment of Related RNA Pairs

We divided the range of scores in the Ribosum-95 matrix into "buckets", where the size of each bucket is 0.01, and the value of each bucket is the probability of a random score falling into it. This gives us the coefficient of a generating function representation of the score distribution of one pair. The polynomial exponents do not start at zero, since a score of -11.4 falls into bucket with exponent -1140. To get the score distribution of a kmer, we simply convolve the polynomial, to raise it to the k^{th} power, keeping in mind the above offsets of exponents.

4.1.2 Alignment of Random Pairs

We assume equal background probabilities of the four nucleotides. From these, we compute probabilities of pairs in random DNA by Pr[pair XY] = Pr[nt X].Pr[nt Y]. Thus we get 16 probabilities for the occurrence of each of 16 pairs. By performing a cross-product operation of this size-16 vector to itself, we generate a rectangular matrix whose cell in location (x, y) gives the probability of alignment of pair x to pair y in random sequence. For the choices of Pr[A] = 0.25, Pr[C] = 0.25, Pr[G] = 0.25, Pr[T] = 0.25, all the probabilities compute to 4^{-k}.

4.1.3 Alignment of RNA Pair to Random Pair

We also generated a probability matrix for the case where a nucleotide pair from a random sequence is aligned to an RNA pair. This is the case of an RNA query being aligned to random DNA that is not a homolog. The probability of a pair A to pair B alignment is the product of probability of pair A in RNA sequence and that of pair B in random sequence. The probabilities of pairs in random sequence are computed exactly as given above. Probabilities of pairs in RNA are computed from Table 3.2.

4.2 Building Static Tables

In this section, we describe algorithms to build tables of possible hit sequences that are independent of the query and target strings. These tables are built only once, after which they could be maintained in memory for use with multiple target and query sequences.

4.2.1 The Pair Neighbour Table

Given a query pair kmer q_i, our key idea rests on identifying a region in the target that forms a pair kmer p such that p is a "close" match to q_i. An alignment is deemed a close match if its score is above a specified threshold, as per the given scoring matrix. For example, using the Ribosum-95 scoring matrix in Table 2.2, the 6-mer AU-GC-GU-GC-CG-CG aligned to AU-CG-GU-CG-CG-AU scores 20.45, while the same 6-mer aligned to UA-UA-UA-GU-GU-UG scores only 6.84. For a threshold $T = 20.0$, the former would qualify as a "hit" while the latter would not (even though both have positive log odds

scores). Given the threshold T and a kmer q, we would like to compute the set S of all kmers that count as hits when aligned with q – we denote this set $N(q)$. Each kmer $t \in N(q)$ is a *neighbour* of the kmer q. Note that, in order to compute the set of all neighbours of kmer q, it is not necessary to know the target sequence or the query. Hence this computation can be performed once at startup, and the neighbourhood of each kmer so computed is stored in the Static Pair Neighbours table. An efficient algorithm to compute neighbourhoods based on a pairwise scoring matrix has been described by Brejová *et al.* [3]. We briefly describe our implementation below.

Algorithm 1 Static Pair Neighbour Table Construction

```
for every kmer q ∈ {AU, CG, GC, GU, UA, UG}^k do
  for every kmer t ∈ {AU, CG, GC, GU, UA, UG}^k do
    if q aligned to t scores above T then
      add t to the set N(q)
```

This brute-force algorithm is straight-forward, but executing it for 6-mers or 7-mers takes an excessive amount of time. We use a simple idea to speed it up significantly. Notice that if the 6-mers $p_1p_2p_3p_4p_5p_6$ and $p_7p_8p_9p_{10}p_{11}p_{12}$ are neighbours, where $p_i \in P$, then it follows that $p_3p_1p_6p_2p_4p_5$ is a neighbour of $p_9p_7p_{12}p_8p_{10}p_{11}$, and vice versa. In fact, given two neighbouring kmers k_1 and k_2, if we permute both of them in the same manner to yield kmers k_1', k_2' respectively, then k_1' and k_2' are neighbours of each other.

We assume an ordering of the set P, *e.g.* AU = 1, CG = 2, ..., UG = 6. Now, each kmer formed from these symbols can be represented numerically, *e.g.* AU-GC-GU-GC-CG-CG = 134322. Note that this 6-mer can be permuted to give the 6-mer AU-CG-CG-GC-GC-GU = 122334, the latter having a lower numerical value than the former. We modify the above algorithm as follows: we only compute the neighbourhood of a pair kmer q if it is in sorted order. If the kmer q is not sorted, we can compute $N(q)$ from $N(q')$ since every neighbour of q has one and only one corresponding entry in $N(q')$. Figure 4.1 shows a conceptual view of the Static Pair Neighbour table.

For a given threshold and matrix, once the kmer neighbourhoods are computed, they can be written to a file. Reading in this file and populating the static tables for 7-mers can be done quickly, given that the sizes of the resulting files are not excessively large. Table 4.1 below gives the sizes of this file, for a range of thresholds for pair 7-mers, using the Ribosum-95 matrix. Note that as the threshold is lowered, the file size increases but is easily handled for the range of thresholds considered (*e.g.* $T = 31.49$).

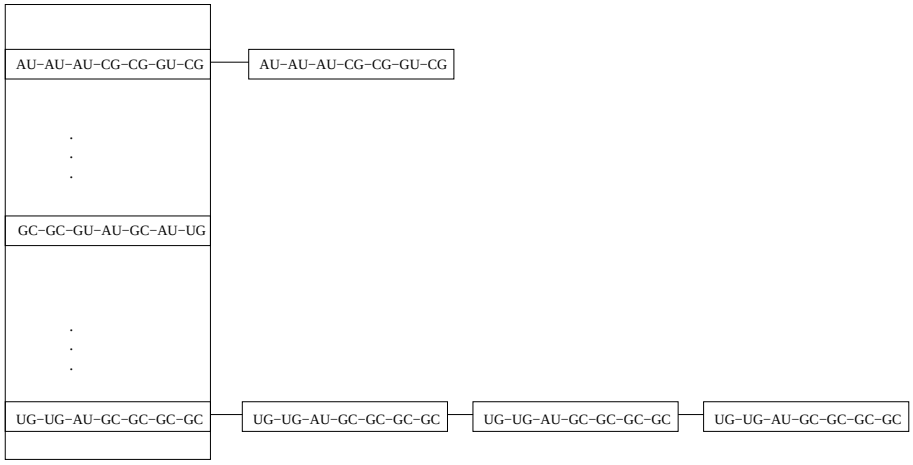

Figure 4.1: Conceptual view of the Static Pair Neighbour table for pair 7-mers with T = 31.49. The pair 7-mer AU-AU-AU-CG-CG-GU-CG has exactly one neighbour, itself. The pair 7-mer GC-GC-GU-AU-GC-AU-UG has no neighbours, that is, there is no pair 7-mer which aligns with this one to score above 31.49. The pair 7-mer UG-UG-AU-GC-GC-GC-GC has three neighbouring 7-mers, as shown. The number of neighbours in these examples is small, but in practice it can be quite large, depending on the threshold. For example, the pair 7-mer GC-GC-GC-GC-GC-GC-GC has 288 neighbours at $T = 31.49$.

Table 4.1: Size of pair 7-mer Static Pair Neighbours file *versus* threshold score

Threshold Score	Hit Rate	File Size (Mbytes)
33.90	2.00%	3.43
33.46	2.64%	3.67
32.79	3.94%	4.31
32.28	5.22%	5.26
31.49	7.73%	8.26
30.85	10.17%	13.08
29.86	14.86%	29.41

Note that the Static Pair Neighbour table is analogous to the use of a hash table of neighbourhoods by a protein homology method that employs scoring matrices.

4.2.2 The Nucleotide Partner Table

Once static pair neighbourhoods have been computed, they have to be converted to a form that allows hashing of pairs in the target sequence. This is because the target is a sequence of nucleotides, not of pair symbols. To this end, we first build a second static table, the Nucleotide Partner table.

We first motivate the need for this table, and then describe how to construct it. Suppose that we are hashing pair 6-mers and using a threshold score of 28.4 (which achieves a 2% hit rate). Then, the Static Pair Neighbour table tells us that the pair 6-mer p_1 = UA-UA-UA-UA-UA-UA has zero neighbours, while the 6-mer p_2 = UA-UA-CG-GC-GC-CG has seven neighbours. That is, there are seven other pair 6-mers which, when aligned to p_2, score above 28.4, and no pair 6-mer exists that scores above 28.4 when aligned to p_1. This implies that if p_2 occurs in a target sequence, the algorithm needs to keep track of its location, in case any given query sequence contains a pair kmer that is one of the seven neighbours of p_2. Similarly, the algorithm can safely ignore all occurrences of the pair kmer p_1 in the target, since no query kmer can align to it and score above the threshold. In this way, we first identify all pair kmers with at least one neighbour. These are the pair kmers whose locations we would like to hash in the target.

The next step is to convert this list of pair kmers into a list of nucleotide kmers. Consider the pair 6-mer p_2 = UA-UA-CG-GC-GC-CG. It is made up of two nucleotide 6-mers, n_{left} = UUCGGC, n_{right} = GCCGAA. Note that, as the names suggest, n_{right} has to occur *after* n_{left} in the sequence, in order for the pair 6-mer p_2 to be possible. Further, note that n_{right} is in *reverse* order in relation to n_{left}, since the pairings among their nucleotides are nested. We denote this situation by saying that UUCGGC is a *partner nucleotide 6-mer* of GCCGAA, since the two nucleotide 6-mers are partners in forming the pair 6-mer p_2. In general, for a pair kmer p made up of n_{left} and n_{right}, we mark n_{left} as being a partner of n_{right}. This is illustrated in Figure 4.2. The static Nucleotide Partner table is built by the following algorithm, where $P(q)$ is the set of partners of the nucleotide kmer q, and $N(r)$ is the set of neighbours of the pair kmer r:

Algorithm 2 Static Nucleotide Partner Table Construction

for every pair kmer $q \in$ Static Pair Neighbour table with $|N(q)| > 0$ **do**
  compute nucleotide kmers n_{left}, n_{right} of pair kmer q
  add n_{left} to the list $P(n_{right})$

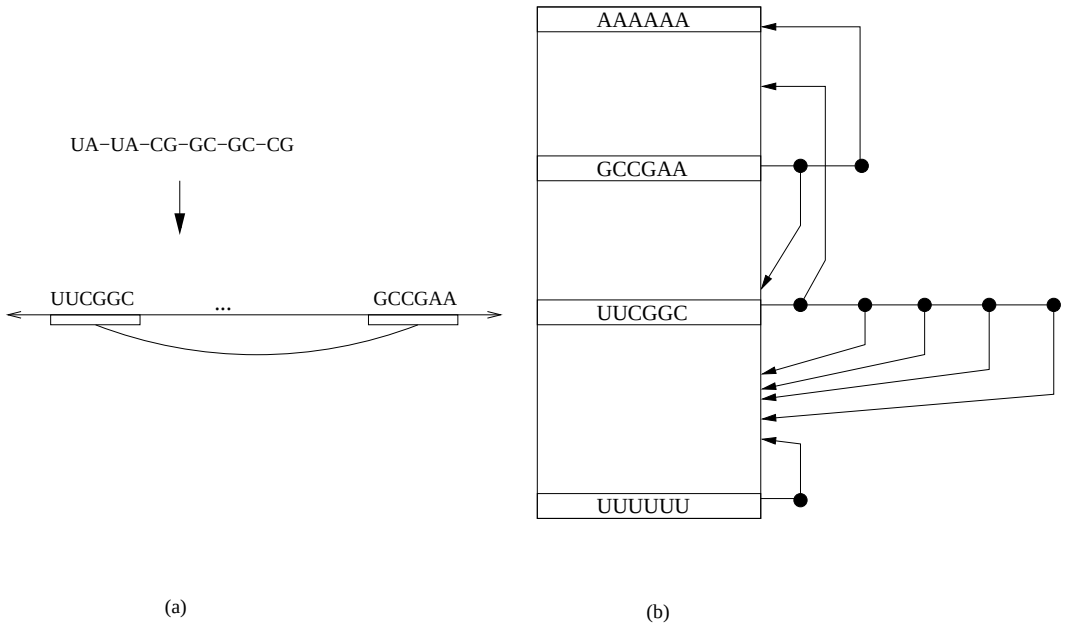

Figure 4.2: The Static Nucleotide Partner table. (a) The pair 6-mer UA-UA-CG-GC-GC-CG and its corresponding nucleotide 6-mers in their linear orientation in a nucleotide sequence. (b) A snapshot of the 6-mer Nucleotide Partner table showing each entry with a list of pointers, where each pointer is to a partner of that entry. For example, the entry "GCCGAA" contains a pointer to its partner "UUCGGC", indicating that they form the pair 6-mer in part (a). Note that partnership relations in this table are 1-way. That is, the entry "UUCGGC" may not necessarily have a partner pointer to the entry "GCCGAA".

Note that for nucleotide kmers n_1, n_2, if n_1 is a partner of n_2, it does not automatically imply the inverse relation, that is, it is possible that n_2 is not a partner of n_1. To understand why, consider the above example. If n_{right} is a partner of n_{left}, it implies that the pair kmer p_2' = GC-CG-CG-GC-AU-AU has at least one pair kmer as a neighbour. However, from Table 2.2, it is seen that AU *versus* AU scores 4.22, while UA *versus* UA scores 4.71. Hence it is possible, for an appropriately chosen threshold, for p_2' to have no neighbours, even though p_2 does.

Finally, we briefly comment on the memory requirements of the static tables in memory. Table 4.2 below shows the combined footprint in memory of the Static Pair Neighbour and Static Nucleotide Partner tables, for a range of thresholds for pair 7-mers. The values have been captured at runtime and indicate that the requirements of these tables are easily within the limits of a typical desktop machine for our prototype implementation.

Note that since these tables are independent of the target and query sequences, their sizes only depend on the threshold score and the scoring matrix.

Table 4.2: Memory requirements of static tables for 7-mers. For example, to hash pair 7-mers, with $T = 31.49$, the algorithm would build and maintain static tables that require less than 12 Mbytes of memory.

Threshold Score	Size in Memory (Mbytes)
33.90	8.29
32.79	8.86
31.49	11.16
29.86	22.56

4.3 Target Database Processing Algorithm

This section describes the algorithm devised to process the unannotated target sequence. The intent of processing the target nucleotide sequence is to identify the locations of high-scoring, potentially *paired* regions, given a threshold score T, a scoring matrix M, and a seed s. These potentially paired regions can then be checked against query paired regions to identify extensible hits. An example of a potential paired region is shown in Figure 4.3.

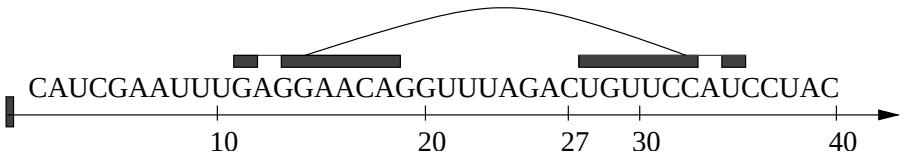

Figure 4.3: A potential pair 7-mer in the target sequence. Using the pair seed 1011 1111, the nucleotide 7-mers GxGGAACA and UGUUCCxU partner together to form the pair 7-mer GU-GC-GC-AU-AU-CG-AU. Hence, the pair kmer GU-GC-GC-AU-AU-CG-AU occurs at position (11, 27), as shown. For a given seed, the left and right positions uniquely identify this potential pair 7-mer among all others in the target sequence.

The outcome of the target processing algorithm is that locations of every high scoring pair kmer are stored in the Pair Hash table. This is analogous to a nucleotide kmer hash table that is used in BLAST or PatternHunter. For a given k, *e.g.* $k = 11$, the BLAST hash table has 4^{11} entries, each entry storing a list of locations of that particular 11-mer.

In the Pair Hash table, there are a total of 6^k entries, where each entry stores the location of that particular pair kmer. The location of a pair kmer, as shown in Figure 4.3, is a tuple of integers (*leftStart, rightStart*) that identify the start positions of the left and right paired regions.

Given a database of size n, and a window size w, there are $\Theta(nw)$ possible pair kmers. However, we expect that most of those pair kmers would be unlikely in RNA sequence, *e.g.* GG-UC-AA-CA-CC-AG-AU, which contains low-scoring pairs such as GG, UC and CC. There is no value in hashing locations and separations of such pair kmers. We would like to identify those pair kmers that are likely to be found in RNA, and furthermore, will score above the specified threshold when aligned to another RNA pair kmer. By building the Static Pair Neighbour table, we already have captured the static list of all high-scoring pair kmers (and their neighbours). Further, from that list, we have identified nucleotide kmers that pair together to form those pair kmers – this is captured in the Static Nucleotide Partner table.

Hence, what remains is to identify occurrences of those nucleotide kmers in the target, and to pair such occurrences together to identify high-scoring pair kmers. In order to achieve this, given a seed s, we have to hash the target nucleotide sequence simultaneously with seed s and its reverse s^R, to find possible pairs. For example, the reverse of the seed 11011111 is 11111011. The seed 1111 is palindromic, that is, its reverse seed is itself. The algorithm is listed below, followed by an explanation.

Algorithm 3 Target Database Processing

```
for target position pos from 1 to n do
    compute lkmer = kmer at pos using seed s
    hash lkmer into Left Nucleotide Kmer table
    compute rkmer = kmer at pos using seed s^R
    for each partner ∈ P(rkmer) in the Static Nucleotide Partner table do
        for each location loc ∈ Locations(partner) in the Left Nucleotide Kmer table do
            if pos is within window size of loc then
                store pair kmer (loc, pos) in Pair Hash table
```

Each position in the sequence is hashed *twice* – once with the pair seed s and again with the reverse seed s^R. We use the convention that hashing any position p with seed s gives the "left" kmer l_p, and hashing it with seed s^R gives the "right" kmer r_p. If the sequence starting at position p participates in a pair kmer, then l_p serves as the left half (pairing with a downstream right half), and r_p serves as the right half (pairing with an

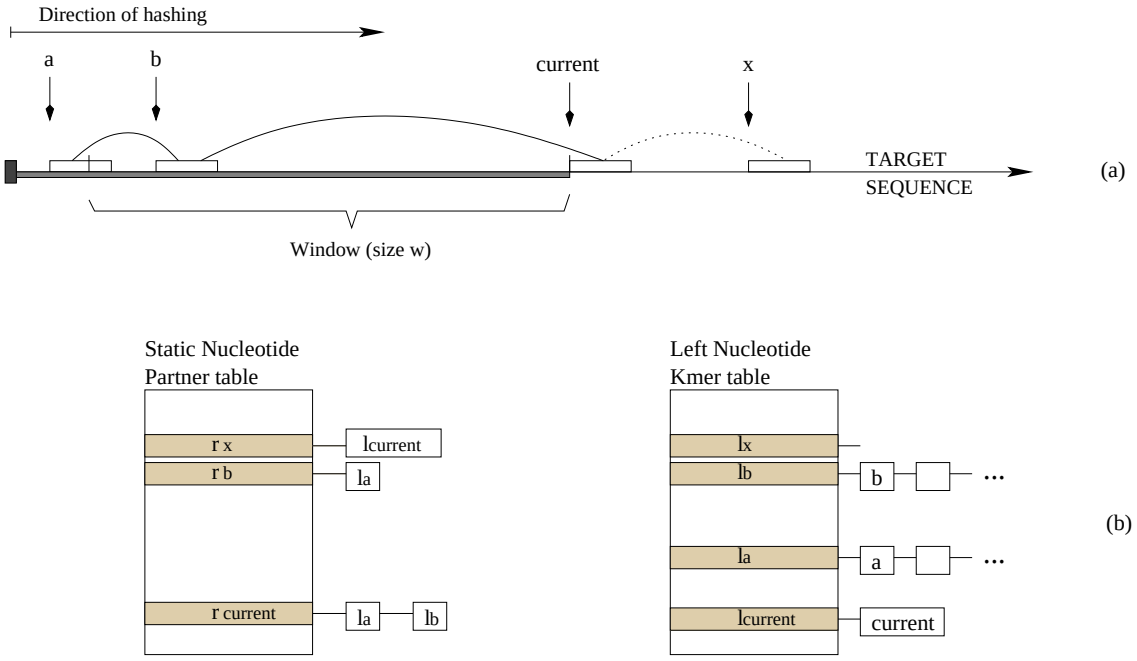

Figure 4.4: Hashing of target nucleotide sequence. The figure is a snapshot at the instant the position *current* is being hashed. (a) Positions a, b form the pair kmer (a, b) as shown. Further, $(a, current)$ and $(b, current)$ are both potential pair kmers. However, the algorithm ignores pair kmers that have an arc greater than the maximum specified window size w. The kmer at position x could also form a high scoring pair kmer $(current, x)$ which is not discovered until the algorithm is at position x. (b) The Static Nucleotide Partner table and the Left Nucleotide Kmer table are shown at the instant that position *current* is being hashed. Note that $l_{current}$ and $r_{current}$ – for *left* and *right* kmers – are the nucleotide kmers at position *current*, using the seeds s and s^R respectively. For details of the algorithm, please see the accompanying text.

upstream left half). See Figure 4.3 for an example.

Now consider Figure 4.4. First, note that before hashing of the target starts, the Static Nucleotide Partner (SNP) table has been fully populated. For example, it identifies the kmers l_a, l_b as partners of $r_{current}$.

When the position *current* is hashed, first the kmer $l_{current}$ is marked in the Left Nucleotide Kmer (LNK) table as occurring in the position *current*. This is to allow discovery of all pair kmers $(current, x)$ later, where $x > current$, as shown. Next, the kmer $r_{current}$ is computed and the SNP table searched for its partners. As l_b is such a

partner in the SNP table, the LNK table is searched for all occurrences of the kmer l_b. One occurrence is found at position b, which is within the window size w from position *current*. Thus, the pair kmer $(b, current)$ is marked in the Pair Hash table. Note that as a result of how we build the LNK table, each list in it is sorted in ascending order. Our algorithm is efficient as it takes advantage of this fact by iterating backwards from the last element, ensuring that only elements within the window size w are processed.

The SNP table also identifies l_a as a partner of $r_{current}$. The LNK table is searched for occurrences of l_a. The closest one is found to be at position a, which is outside the window allowed. Hence, the algorithm concludes that $(a, current)$ is not a valid pair kmer.

This completes the hashing at position *current*. The algorithm moves to position $current + 1$ and repeats the above procedure. When at position x, it is found that r_x has the partner $l_{current}$ in the SNP table. Further, $l_{current}$ is seen (in the LNK table) to occur at position *current*, which is within the window at position x. Thus, the pair kmer $(current, x)$ is discovered. Further, if position b also participated in a pair kmer as the *right* half, such a kmer *e.g.* (a, b) is discovered when position b is hashed, as shown in Figure 4.4. In this way, all high scoring pair kmers in the target sequence are discovered in a single pass through the nucleotide sequence, and recorded in the Pair Hash table.

Figure 4.5 shows a conceptual view of the Pair Hash table. It is analogous to a nucleotide hash table which stores, for every kmer, the positions in the target where it occurs. Simlarly, the Pair Hash table stores, for each pair kmer, the location*s* of the *left* and *right* paired segments that make up the pair kmer.

If we are given a target sequence of size n, and a window size w, there are $\Theta(nw)$ possible pair kmers that need to be hashed. If for every target position, we examined a window of size w around it, the algorithm would take $\Theta(nw)$ time, which is quite inefficient even though it is linear in n (for constant window size). The above algorithm is much more efficient on average, since it only looks "back" among the w positions from *current* if there are high-scoring nucleotide kmers there.

Finally, the Pair Hash table, once built, contains all the information necessary to find high-scoring hits between query and target pair kmers. After the target is hashed, the LNK table is deleted since its purpose is only to aid in creation of the Pair Neighbour table.

Once the algorithm has completed, the Pair Hash table is fully built. We recorded the

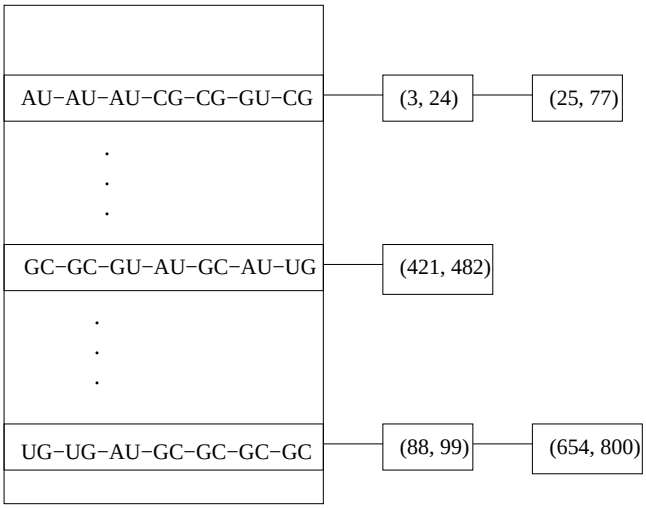

Figure 4.5: Conceptual View of the Pair Hash table. For each pair kmer, the locations in the target where it occurs are stored. Note that each location record is made up of two integer values – the positions of the *left* and *right* segments that make up the pair kmer. For example, the pair 7-mer GC-GC-GU-AU-GC-AU-UG occurs in the target once, with its left segment starting at position 421, and the right at position 482. The lengths of these segments is the length of the seed with which this table was hashed, 7 in this case.

size of the Pair Hash table for all the target sequences used in our experiments. These sizes in memory were captured at runtime, for targets ranging from 7 to 258 Kbases. We found that, over that range of target sizes, the table varies in size from 8.4 to 8.8 Mbytes, indicating that it can be contained within an average desktop machine's RAM. For the entire human chromosome 22 sequence (47 Mbases), the resulting Pair Hash table is of size 47.5 Mbytes (for 7-mers with $T = 31.49$), easily contained within a desktop machine's RAM. There is one interesting difference between a nucleotide hash table and the Pair Hash table: two different targets t_1, t_2 of the same length would result in nucleotide hash tables of the same size; however, targets t_1, t_2 of the same length may result in Pair Hash tables of different sizes, depending on their nucleotide content. For instance, a target sequence of all A's (*e.g.* AAAAAA...), regardless of its length, would result in zero entries in the Pair Hash table.

Given the nucleotide sequence and secondary structure strings of the query, as for Figure 2.1, it is a trivial matter to parse them and extract all pair kmers in the query. Figure 4.6 is an example of a query sequence and structure. The corresponding list of unpaired and paired regions that result from parsing it are shown in Table 4.3.

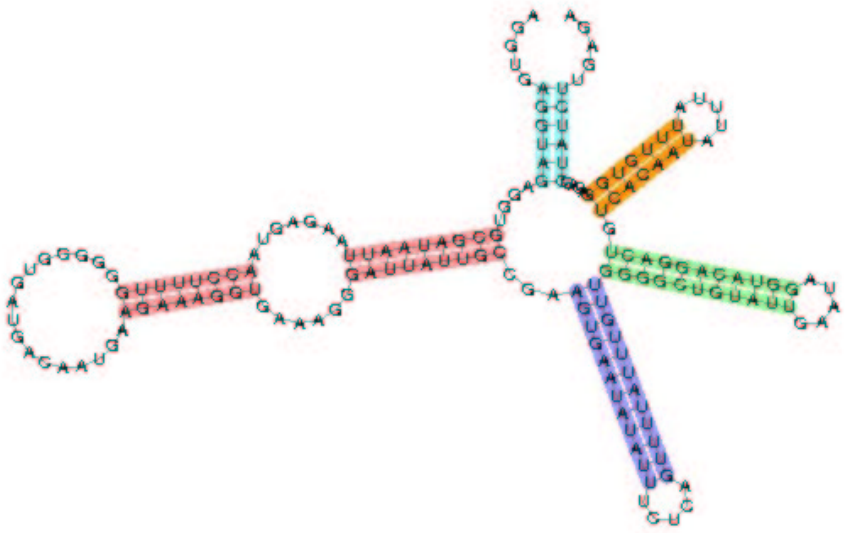

Figure 4.6: Lysine Riboswitch secondary structure

Table 4.3: Paired regions of Lysine Riboswitch

No.	Left start	Right Start	L/R Segment Size
1	31	56	8
2	15	70	9
3	82	104	12
4	116	133	11
5	145	157	7
6	5	168	6

4.4 Algorithm to Find Pair Hits to a Given Query

We have shown in Section 4.3 how the target nucleotide sequence is processed to yield a hash table of locations of pair kmers and how a given query sequence is resolved into pair kmers. We must find hits between the query and target pair kmers.

4.4.1 1-hit Method – 1 Pair 7-mer

As mentioned previously, a hit is an alignment of a query pair kmer and a target pair kmer that scores above a predefined threshold score. In the 1-hit pair 7-mer method, we seek to find a pair 7-mer in the target that is a high-scoring hit to any pair 7-mer in the query. When such a hit is found, a "window" around the hit is marked to identify where the query would be located if this hit is a true one. The algorithm to find pair hits in the target is straight-forward and given below:

Algorithm 4 1-hit Target Hit Finding (1 Pair Kmer)

for each query pair kmer qk from qk_1 to qk_n **do**
  for each $neigh \in N(qk)$ in Static Pair Neighbours table **do**
    for each pair kmer $pk \in Locations(neigh)$ in Pair Hash table **do**
      if $|arc(qk) - arc(pk)| \leq$ arc margin **then**
        pk is a valid hit to qk

Briefly, this algorithm iterates over all the pair kmers of the query. For each query kmer qk, it looks in the SPN table to find its neighbouring kmers, that is, kmers that score above T when aligned with qk. Assume that $neigh$ is one such kmer. Then, the Pair Hash table contains all locations in the target of the kmer $neigh$. Hence, the algorithm finds the target kmer pk at each location of the kmer $neigh$, checking if the arc of pk is close to that of qk – if so, a hit has been found. For example, if $qk = (7, 20)$ and $pk = (1800, 1844)$, then the $arc(qk) = 13$ and $arc(pk) = 44$. For an arc margin of 10 nucleotides, these are too far apart to be considered a hit. Note that in our implementation, entries of the form $p = (l, r)$ in the Pair Hash table are sorted in increasing order of r. But entries with the same value of r are not sorted by l. If that were so, then identifying pair kmers with close arc values could be done more efficiently – we identify this for future work.

We could have chosen to find hits differently, by iterating over each entry in the Pair Hash table while checking for matches to query pair kmers. We have chosen our current

implementation since we expect on average the number of query kmers will be smaller than the number of entries in the Pair Hash table.

4.4.2 2-hit Method – 2 Pair 4-mers

We also investigated a 2-hit approach in which two smaller pair hits are required (specifically, pair 4-mers) instead of a single pair 7-mer hit. We expect this approach to work better for RNA queries that have paired content that is short and discontinuous. Figure 3.4 shows how the two pair hits could be oriented with respect to each other. The algorithm for finding an extensible hit using this approach is given below. Note that this algorithm uses the above one to find hits to a single pair kmer. In addition, this algorithm has to verify that two candidate hits have the same relative orientation as the query pair kmers they correspond to. Lastly, the "separation" of query kmers has to be within a "margin" of that of the target kmers.

Algorithm 5 2-hit Target Hit Finding (2 Pair Kmers)

```
for each set of two disjoint query pair kmers (qk1, qk2) do
  querySep = separation of qk1 and qk2
  qk1Hits = find hits to qk1 by method in Sec.4.4.1
  qk2Hits = find hits to qk2 by method in Sec.4.4.1
  for each hit hit1 ∈ qk1Hits do
    for each hit hit2 ∈ qk2Hits do
      if hit1, hit2 are overlapping, pseudoknotted, or in wrong orientation then
        they cannot form a valid hit
      else
        targetSep = separation of hit1 and hit2
        if |targetSep − querySep| ≤ separation margin then
          (hit1, hit2) is a valid hit to (qk1, qk2)
```

4.4.3 2-hit Method – 1 Pair 4-mer + 1 Nucleotide 8-mer

We also experimented with a 2-hit approach that requires one hit to a (shorter) pair 4-mer and a second hit to a nucleotide 8-mer. This approach is similar to the case of using two pair hits, except that since one hit is a nucleotide hit, a separate nucleotide hash table is required (in addition to the Pair Hash table). Also, note that only two relative orientations of the two hits are possible – shown in Figure 3.5. The two above methods

find extensible hits solely based on paired regions, which could be a disadvantage if a query RNA sequence has low paired content. In contrast, this method takes into account both the paired and unpaired content in the query, and may perform better on RNA query sequences with low paired content.

The algorithm to find an extensible hit using this approach is given below.

Algorithm 6 2-hit Target Hit Finding (1 Pair Kmer + 1 Nucleotide Lmer)

for each set of two disjoint query pair and nucleotide kmers (q_p, q_n) **do**
  $queryOrient$ = orientation of q_p w.r.t. q_n
  $querySep$ = separation of q_p and q_n
  $qpHits$ = find hits to q_p by method in Sec.4.4.1
  $qnHits$ = find hits to q_n using a nucleotide kmer hash table
  for each hit $hit_p \in qpHits$ **do**
    for each hit $hit_n \in qnHits$ **do**
      if hit_p, hit_n are overlapping or in wrong orientation **then**
        continue
      $targetSep$ = separation of hit_p and hit_n
      if $|targetSep - querySep| \leq$ separation margin **then**
        (hit_p, hit_n) is a valid hit to (q_p, q_n)

Note that since our goal is to evaluate the sensitivity and specificity of the three methods presented above, the 2-hit algorithms presented above are not optimal in terms of runtime performance. Typically, 2-hit methods can be implemented by sorting each seed hit based on its "diagonal", that is, a value related to $(x_t - x_q)$, where x_t and x_q are the target and query positions of the kmers that form the hit, respectively. In the next step, hits in the same (and possibly "nearby") diagonal lists are compared to find valid *pairs* of hits. In this work, we have instead chosen a simpler implementation for 2-hit methods that partitions the target sequence into smaller blocks and pair-hashes each partition in turn, allowing a big enough overlap between adjacent partitions to accommodate the largest window size of a pair kmer. This avoids having to build additional data structures to store details of hits by diagonals, and a reference from each hit to its query pair kmer. As we note later, a future area of work would be to investigate data structures that achieve more efficient runtime performance of the 2-hit methods, possibly by sorting hits by diagonal.

Chapter 5

Experiments and Results

Based on the theoretical specificity and sensitivity computations presented in previous sections, we have arrived at hypotheses about the performance of pair seeding in detecting homologs of a single RNA molecule in a given target sequence. In this section, we first list the hypotheses whose validity we want to evaluate. Then, we describe the experiments to perform such an evaluation. All experiments use the Ribosum-95 scoring matrix.

5.1 Hypotheses

1. The 1-hit pair 7-mer approach, with a suitable threshold, delivers the same sensitivity and higher specificity than the BLAST size-11 nucleotide seed. This is for a small arc margin between the query pair kmer and hit. The runtime of this method will be higher than for a BLAST-seed approach, but comparable for high thresholds.

2. The 2-hit approach using 1 pair 4-mer and 1 nucleotide 8-mer will achieve better specificity than BLAST for the same sensitivity, for small arc margin and separation margin values. Further, for queries with a lower proportion of paired content, this method will outperform the 1-hit pair 7-mer approach in sensitivity, as it does not ignore unpaired query sequence. The runtime here will be considerably slower than that of the 1-hit method in our current implementation. (For a more efficient version that sorts hits by arc and separation margins, it may be possible to approximate the runtime of the 1-hit method, but we leave this for future work.)

3. The 2-hit approach using 2 pair 4-mers will give lower false positives than the BLAST seed as well as the 1-hit 7-mer approach, for small arc margin and separation margin values. However, this method will do poorly on queries with low paired content. This method also involves more processing than the 1-hit approach so will have high runtimes.

5.2 Experimental Setup

5.2.1 Sensitivity and Specificity Estimation

All experiments are based on data from the RFAMSEQ [9] database of nucleotide sequences; each RNA family within the RFAM database has homologs identified in RFAMSEQ. RFAM captures the member sequences of each RNA family as a multiple sequence alignment (hand-curated by the creators of RFAM). One such alignment is shown in Table 5.1, for the *let-7* microRNA precursor family (RF00027 in RFAM). Below the alignment of nucleotide sequences is the consensus secondary structure of this molecule in bracket notation.

For example, the second member in the alignment occurs in a human sequence with the EMBL Accession# AL158152.18. Note that the *let-7* RNA family has three members that are from this EMBL entry, located in the intervals 40779 – 40863nt, 38291 – 38377nt and 37901 – 37981nt.

We treat a particular sequence/structure as the query, and we execute a homology search using a given seeding method, *e.g.* 1-hit pair 7-mers with $T = 31.49$, over the target sequence AL158152.18. We repeat the search using the same query, with all other target sequences in the multiple sequence alignment one by one. Then, we repeat this process for each query in the family, effectively searching for every query in every target.

In each homology search, a window for hit extension is identified around each seed hit. The window size is set to the length of the query sequence taking into account the relative position of the seed hit within the query. Since we assume that the RFAM annotation contains all true hits, we can infer that any window identified outside the range of the known true hits is a false hit.[1]

[1]Note that the RFAM database is annotated using a BLAST search of member sequences. Hence, it is quite possible in practice that an EMBL database file has "real" homologs of a RFAM family that are

The more true hits and the fewer false hits an algorithm marks, the better it is deemed to have performed. Consider the RF 265 family as an example. It has three query sequences, each of which occurs in a target sequence. Each target has a reverse-complement version, giving eighteen searches. Since we do not double-count hits – if q_1 hits t_2 and q_2 hits t_1, we count that as 1 true hit – this gives a total of nine true hits for this family. Over the eighteen executions, the total target sequence length is 2,069,322 nucleotides. Of these, we count all falsely marked nucleotides – these are the false hits for that particular method.

As can be seen, due to the small number of true hits in our experiments, statistical significance is lacking in those results. However, false hits are counted in nucleotides and can amount to hundreds of thousands of nucleotides, as shown in Section 5.4. When executions are over megabases of target sequence, false hits can be used to compare the various methods. To this end, we performed false hit testing on the human chromosome 22 sequence, as detailed later.

5.2.2 Runtime Estimation

There are two components to the runtime of a pair hashing method. The first relates to the time it takes to hash a target sequence into pair kmers. As seen in Section 4.3, hashing pair 7-mers involves additional processing, as compared to the simpler method of hashing nucleotide 11-mers (as in BLAST). We capture the time to pair-hash database sequences of various sizes, to determine how it compares to the time to hash the same target using a nucleotide spaced seed.

The second component of runtime refers to actually finding hits to a given query, once the target has been pair hashed. This involves the time to process the query into pair kmers, and to find high-scoring hits to each kmer in the Pair Hash table. For a 1-hit approach, this time is expected to be very small. For a 2-hit approach, there is additional processing involved in combining individual hits of smaller seeds to form extensible hits.

not annotated in RFAM, since a BLAST search misses them. However, to identify such hits as homologs requires a biologist's expertise in RNA structures. Hence for the purposes of this study, we make the assumption that the annotation of homologs in RFAM is complete.

Table 5.1: Seed multiple sequence alignment of the *let-7* microRNA precursor family. Note that the second block gives sequences that follow those in the first block.

AP001359.4/114553-114478	Hom.sap.	CCAGGUUGAGGUAGUAGGUUGUAUAGUUU..A.GA.....AUUACA.UC.....AAGGGAGAUAACUGUACAGCCUCCU
AL158152.18/40779-40863	Hom.sap.	CUAGGAAGAGGUAGUAGGUUGCAUAGUUUU.AGGGC.AGGGAUUUUGCCC...ACAAGGAGGUAACUAUACGACCUGCU
AL158152.18/38291-38377	Hom.sap.	UCAGAGUGAGGUAGUAGAUUGUAUAGUUGU.GGGGU.AGUGAUUUUACCC.UGUUCAGGAGAUAACUAUACAAUCUAUU
AC048341.22/3536-3622	Hom.sap.	CCUGGCUGAGGUAGUAGUUUGUGCUGUUGGUCGGGU.UGUGACAUUGCCC..GCUGUGGAGAUAACUGCGCAAGCUACU
AL049853.1/18009-17931	Hom.sap.	UUGGGGUGAGGUAGUAGGUUGUAUAGUUUG..GGGC.......UCUGCCC..UGCUAUGGGAUAACUAUACAAUCUACU
AC094021.2/47340-47253	Hom.sap.	CCAGGCUGAGGUAGUAGUUUGUACAGUUUG.AGGGUCUAUGAUACCACCC.GGUACAGGAGAUAACUGUACAGGCCACU
AC018755.3/119936-120011	Hom.sap.	CCGGGCUGAGGUAGGAGGUUGUAUAGUU....GAGG....AGGACACC.C.....AAGGAGAUCACUAUACGGCCUCCU
AL049853.1/17069-16987	Hom.sap.	.CGGGGUGAGGUAGUAGGUUGUGUGGUUUC.AGGGC.AGUGAUGUUGCCC...CUCGGAAGAUAACUAUACAACCUACU
AL158152.18/37901-37981	Hom.sap.	.UGGGAUGAGGUAGUAGGUUGUAUAGUUUU.AGGGU.......CACACCCACCACUGGGAGAUAACUAUACAAUCUACU
AP000962.2/118093-118168	Hom.sap.	CCGGGUUGAGGUAGUAGGUUGUAUGGUUU..AGAGU.......UACACCC.....UGGGAGUUAACUGUACAACCUUCU
AF210771.1/1883-1955	Cae.bri.	UACGGU.GAGGUAGUAGGUUGUAUAGUUU..AGAAU.......AUUACUC.....UCGGUG..AACUAUGCAAGUUUCU
Z70203.2/12451-12379	Cae.ele.	UCCGGU.GAGGUAGUAGGUUGUAUAGUU....UGG....AAUAUUA.CCA.....CCGGUG..AACUAUGCAAUUUUCU
SS_cons		<<<<<<.<<<<<<<<<<<<<<<<<<<<....<<<<..........>>>>.............>>>>>>>>>>>>>>>
AP001359.4/114553-114478	Hom.sap.	AGCUUUCCUUG
AL158152.18/40779-40863	Hom.sap.	GCCUUUCUUAG
AL158152.18/38291-38377	Hom.sap.	GCCUUCCCUGA
AC048341.22/3536-3622	Hom.sap.	GCCUUGCUAGU
AL049853.1/18009-17931	Hom.sap.	GUCUUUCCUGA
AC094021.2/47340-47253	Hom.sap.	GCCUUGCCAGG
AC018755.3/119936-120011	Hom.sap.	AGCUUUCCCCA
AL049853.1/17069-16987	Hom.sap.	GCCUUCCCUG.
AL158152.18/37901-37981	Hom.sap.	GUCUUUCCUAA
AP000962.2/118093-118168	Hom.sap.	AGCUUUCCUUG
AF210771.1/1883-1955	Cae.bri.	ACCUCACCGAA
Z70203.2/12451-12379	Cae.ele.	ACCUUACCGGA
SS_cons		>>>>>>>>>>>

5.3 Experiments to Validate Hypotheses

We describe several experiments below, each one evaluating a different scheme of identifying an extensible seed hit. The list of RFAM families used in the experiments is given in Table 5.2. The table also gives a brief description of the RNA family along with its Family ID in the RFAM database. These families were chosen primarily because they have non-zero number of paired regions of length 7 or more. This feature allows the 1-hit pair 7mer method to be executed on all of them, as well as the remaining two 2-hit methods, so that the three approaches can be compared. As we noted in Section 3.7, it is rare to find pairings of length much greater than 7 in RNA families.

5.3.1 Experiment 1 – 1-hit Pair 7-mer Approach

This experiment tests the performance of the 1-hit approach using a single pair 7-mer hit. Note that this method can be applied only to RNA families that have paired regions of length 7 or more.

Every query q_i is associated with a single target sequence t_i, as described above. Thus, we identify up to n target sequences $t_1, \ldots, t_m,\ m \leq n$ that correspond to the queries. With query q_1, we execute a search in all targets $t_1, \ldots, t_m$ using the 1-hit pair 7-mer as a seed hit. We repeat this procedure for all n queries. In each case, we record the lengths and locations of windows marked by the algorithm for extension, and whether they cover the known true hits. We also note the runtime required to hash the target as well as find hits. We repeat this procedure for all families listed in Table 5.2. This gives us a summary of the performance of the 1-hit pair 7-mer approach, in terms of sensitivity and specificity, as well as runtime.

5.3.2 Experiment 2 – 2-hit Approach with Pair and Nucleotide Seeds

Experiment 1 above tested the performance of the 1-hit pair 7-mer approach and compared it to that of the nucleotide BLAST seed. In this experiment we test a 2-hit approach. A hit is deemed to be extensible only if it involves two smaller non-overlapping hits that are as "close" to each other as their corresponding query kmers are, within an acceptable margin. In particular, an extensible hit here is a combination of one pair 4-mer hit and one nucleotide 8-mer hit of the seed 1111 1111. Closeness is determined by two

Table 5.2: RFAM families used in experiments. The *Size* field shows the average length of the member sequences of that family. The *Members* field is the number of sequences in that family. The $7-mers$ field show the number of different (possibly overlapping) pair 7-mer regions in the molecule. These values are also based on the *consensus* structures of each family and hence are averages. It is possible for a particular member RNA sequence to deviate from these values.

Rfam#	Family	Size	Members	7-mers	Other Details
84	CsrC RNA	255nt	14	11	Carbon storage reg. C ncRNA of *E. coli*, has 1 long, many short paired regions.
103	mir-1 microRNA	77nt	15	10	Simple structure with a longer paired region. Occurs in drosophila, nematode, human.
131	mir-30 microRNA	72nt	7	11	Like mir-1, but mammalian, with a small discontinuity in structure.
235	Plasmid RNA III	131nt	9	13	Occurs in various bacterial plasmids, *e.g. E. faecalis*, with a branched, more complex structure.
265	snoRNA U69	132nt	3	14	Occurs in human and mouse, one of many small nucleolar RNAs, with a linear structure.

criteria: Firstly, the relative orientation of the pair hit and nucleotide hit has to be the same as that of their corresponding queries. A nucleotide kmer can be positioned before, inside or after the pair kmer, as shown in Figure 3.5 of the previous chapter. Secondly, the distance between the nucleotide 8-mer and pair 4-mer in the query should be "close" to that in the target. This has been described in the previous chapter in more detail. We expect that for queries with shorter paired regions, such as those in the RF 84 family, this method would perform better than the one in Experiment 1.

This method has the advantage over the 1-hit method that it uses the information in the unpaired regions of the RNA molecule. Further, since several RNA families do not have long, uninterrupted paired regions, using a shorter pair seed allows the use of pair

seeding on those families. The intent is to achieve higher specificity than a nucleotide seed by combining it with a pair seed, as the latter better discriminates RNA regions from non-RNA ones.

We followed the same procedure as detailed in Section 5.3.1 to collect statistics about true and false positives and runtime.

5.3.3 Experiment 3 – 2-hit Pair 4-mers

This experiment evaluates a 2-hit approach that requires both hits to be pair 4-mers. The two hits are required to be non-overlapping. There are four possible orientations of pair kmers k_1, k_2 with respect to each other: two arise from k_1 being nested inside k_2 or vice versa, and two more from k_1 being before k_2 in the nucleotide sequence, and vice versa. (Note that we do not consider the pseudo-knotted orientation for this work; please see Figure 3.4.) Hence, this method has to ensure that the two hits are oriented in the same manner as their corresponding query kmers. This requires additional processing to evaluate each pair of hits, and consequently a larger runtime than the 1-hit method. This approach, for a properly chosen threshold, can theoretically perform better than a 1-hit pair 7-mer approach. However, it has the disadvantage of ignoring all query sequence information that is in unpaired regions, as it does not require a hit of a nucleotide spaced seed.

5.3.4 Experiment 4 – BLAST and PatternHunter [13] Seeds

We also repeat homology searches using the RFAM queries and targets in Table 5.2 using the BLAST 11-mer seed 111 1111 1111 and the optimal PatternHunter [13] seed of weight 12 and length 16, 1101 1101 0110 1111. The results of using these pure nucleotide seeds are compared to those from the above three experiments, to determine if the pair seed based methods are superior to these or not.

5.3.5 Experiment 5 – Varying Arc Margin

We also repeated the experiments using the same queries and targets as above, this time varying arc margins of pair hits to determine the variation in false hits. As given in Section 3.4, the false positive are expected to vary as $O((nwq)^2ar)$, or linearly with the

separation margin, and linearly with the arc margin for the 2-hit Pair and Nucleotide Kmer approach.

5.3.6 Experiment 6 – Human Chromosome 22

Using all the queries from the five RFAM families used in the above experiments, we ran homology searches on the complete 48 Mb human chromosome 22 target sequence. Since none of the queries used are known to have homologs within this sequence, any hits were deemed to be false positives. We performed this experiment to identify the differences in false positive rates between the various methods identified by execution on a large, multi-megabase sequence, which are more statistically significant than those based on smaller target sequences.

5.4 Results

We executed the various homology methods – 1-hit, 2-hit, 2-hit pair + nucleotide, PatternHunter and BLAST – outlined in the last section with specific threshold scores and seed sizes. Table 5.3 below gives the list of search methods used (for a total of nine), and the seeds and thresholds used in each case. The label of each method in Table 5.3 is used later in this section to refer to each method.

5.4.1 Runtimes

Before discussing the sensitivity and specificity findings, we briefly present the experimentally determined runtimes for hashing of target database sequences.

Figure 5.1 shows the captured runtimes for several nucleotide sequences over a range of lengths from 7,000 to 250,000 bases. Each target sequence was hashed in three diffferent ways: (1) using a nucleotide hashing method with the BLAST weight-11 seed, (2) hashing pair 4-mers and nucleotide 8-mers, and (3) hashing pair 7-mers. The results show that hashing only nucleotides is the fastest method. This is expected since it takes $\Theta(nk)$ time, $k = 11$, while hashing pair 7-mers takes $\Theta(nwk)$, $k = 7$, $w = 300$, as per the theoretical runtime estimates we presented in Section 3.4.

Hashing 7-mers is more expensive than hashing 4-mers mainly due to the larger seed size. In addition, the factor $(6/16)^k f$ in the runtime is influenced by the threshold T as

Table 5.3: Homology methods used in experiments

No.	Homology Search Method	Label
1	1-hit : Pair 7-mers with $T = 29.86$, arc margin = 10nt. $T = 29.86$ is among the lowest usable thresholds for pair 7-mers.	7mer(29.86)
2	1-hit : Pair 7-mers with $T = 31.49$, arc margin = 10nt. $T = 31.49$ is a suitable threshold for pair 7-mers.	7mer(31.49)
3	1-hit : Pair 7-mers with $T = 32.79$, arc margin = 10nt. $T = 32.79$ is too high to use in practice. It is here for comparison with other thresholds.	7mer(32.79)
4	2-hit : Pair 4-mer with $T = 19.6$ + nucleotide 8-mer, arc margin = 5nt, separation margin = 3nt	NP(19.6)
5	2-hit : Pair 4-mer with $T = 19.1$ + nucleotide 8-mer, arc margin = 5nt, separation margin = 3nt	NP(19.1)
6	2-hit : 2 Pair 4-mers with $T = 19.6$, arc margin = 5nt, separation margin = 3nt	PP(19.6)
7	2-hit : 2 Pair 4-mers with $T = 19.1$, arc margin = 5nt, separation argin = 3nt	PP(19.6)
8	Optimal length-16 weight-12 spaced seed 1101 1101 0110 1111	PH
9	BLAST weight-11 seed 111 1111 1111	BLAST

well as the seed size k, as described in Section 3.4. We used $T = 31.49$ for 7-mers, and $T = 19.6$ for 4-mers. From Table 3.5, we note that $f = 0.34$ for $T = 31.49$. That is, for a given relevant 7-mer, there is a 34% chance that it will have at least one neighbour, and hence be stored in the Pair Hash table. Similarly, for $T = 19.6$, we computed $f = 0.12$. Hence, for an arbitrary relevant 4-mer, there is only a 12% chance that it will be hashed into the Pair Hash table. These results demonstrate that our implementation of pair hashing runs in linear time. However, since our implementation is not optimized, it is quite likely that the constant factors in our linear-time algorithm could be reduced.

For the target sizes in Figure 5.1, it is seen that the hashing step takes several tens to hundreds of milliseconds. The next step to find hits is extremely fast for nucleotide seeding based (*e.g.* BLAST) approaches, as well as for the 1-hit pair 7-mer approach. For both these methods, this step is seen to take tens of microseconds or less. This is expected as per theoretical estimates of both steps taking $\Theta(q)$ time. However, in

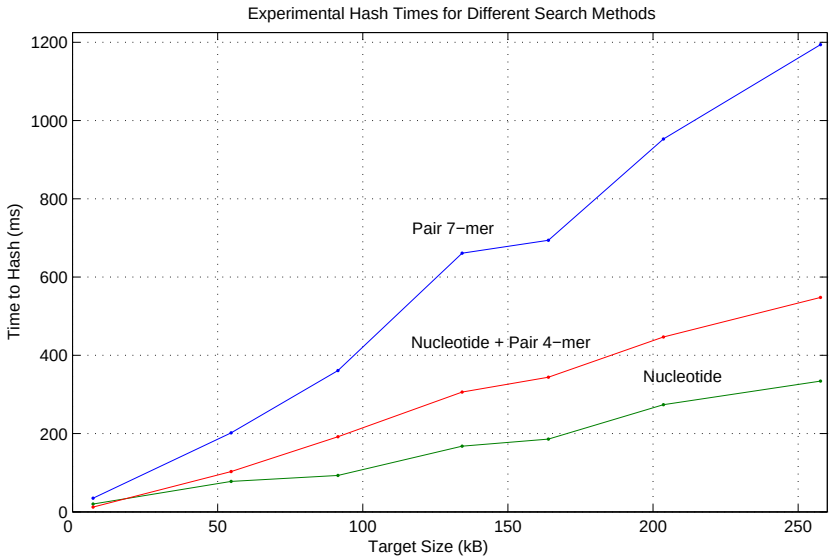

Figure 5.1: Times to hash target nucleotide sequence using different methods. Hash times in milliseconds are shown for three methods – (1) simple nucleotide hashing as in BLAST with a weight-11 seed, (2) pair 7-mer hashing, and (3) nucleotide 8-mer hashing + pair 4-mer hashing. It is seen that pair 7-mer hashing is the most expensive operation. Further, all three methods take linear time to hash.

the 2-hit Pair-Nucleotide method, the hit-finding step takes tens of milliseconds, that is, approximately the same order of magnitude as the hashing step! As mentioned earlier, our implementation is not optimal and can potentially be improved. Nevertheless, these values illustrate how a 2-hit method typically spends considerably more time than a 1-hit method in the hit finding step.

5.4.2 Sensitivity and Specificity

We now present the results of sensitivity and specificity values achieved by the pair hashing based methods. Table 5.4 below summarizes the true positives, and Table 5.5 summarizes the false positives, achieved by each of the nine methods listed in Table 5.3.

For the RF 84 family, a total of 14 members were used as queries, with the EMBL database files they occur in (and their reverse complement versions) as the targets. This gives a total of 105 true hits. Member sequences of this family have lengths of approximately 245nt, but there is a single member that is only about 40nt long. Since this one

Table 5.4: Summary of experimental results: true hits. The last row, for the *BLAST* method, gives the total number of true hits for each RNA family, where each true hit is a nucleotide window of the size of the RNA homolog. The values in other rows show how many of those true hits are successfully found by each method. For example, the *7mer(32.79)* method finds only 21 of 45 true hits for the RF 235 family.

Method	RF 84	RF 103	RF 131	RF 235	RF 265
7mer(29.86)	91	72	27	45	6
7mer(31.49)	91	58	27	45	6
7mer(32.79)	91	45	9	21	6
NP(19.6)	99	120	27	0	6
NP(19.1)	99	120	27	45	6
PP(19.6)	77	3	3	0	6
PP(19.1)	77	13	27	45	6
PH	103	120	27	45	6
BLAST	103	120	27	45	6

Table 5.5: Summary of experimental results: false hits. Each entry in this table is in nucleotides. For example, over all the RF 265 target sequences searched, of total length 2,069,322 nucleotides, the *BLAST* method marks 11,774 nucleotides incorrectly as hits. In comparison, the *7mer(31.49)* method marks only 923 nucleotides in the same targets incorrectly.

Method	RF 84	RF 103	RF 131	RF 235	RF 265
7mer(29.86)	1,703,356	144,464	20,412	83,838	2,110
7mer(31.49)	504,130	34,158	10,384	29,112	923
7mer(32.79)	198,634	10,950	2,576	7,860	661
NP(19.6)	625,747	1,980	5617	0	5,152
NP(19.1)	1,147,188	13,703	9,381	25,764	8,441
PP(19.6)	111,342	307	288	0	17,785
PP(19.1)	488,780	1,827	5,862	234	20,022
PH	369,906	34,539	2,382	24,085	5,403
BLAST	819,083	100,825	13,830	62,383	11,774
Total	53,762,492	49,666,350	9,870,882	3,390,624	2,069,322

member is so much shorter, and contains no pair 7-mers at all (as shown in Appendix A), the 1-hit pair 7-mer approach simply does not work on it. Hence, the low true positives as shown in Table 5.4 for the *7mer(29.86)*, *7mer(31.49)* and *7mer(32.79)* methods.

However, since all queries contain pair 4-mers, the $PP(19.6)$ method works well, giving close to 100% sensitivity with specificity better than the *BLAST* method. However, the *PH* method gives the best sensitivity and specificity.

For the RF 103 family, there are 15 query sequences employed to search 16 targets, to give a total of 120 true hits. In this case, the 1-hit pair 7-mer method has poor sensitivity at the thresholds tested. On further scrutiny (shown in Appendix A), we see that two query sequences have only one pair 7-mer each that aligns with itself to score above the threshold! Note that such a situation can be detected at runtime by processing the query sequence and structure. The $NP(19.6)$ method works best in this case, as seen in Table 5.4. The *PH* method also does well. The $PP(19.6)$ and $PP(19.1)$ methods have poor sensitivity – this is because most queries have only one pair 4-mer that aligns with itself to score above the threshold. This is clearly inadequate since a 2-hit method requires that at least *two* query pair 4-mers exist, that can align to target 4-mers and score high. Again, such a situation can be detected prior to performing the search by analysing the query sequence (as we have done, shown in Appendix A). Once it is determined that the query has only one high-scoring pair 4-mer, the obvious inference is that a 2-hit method requiring two pair 4-mers hits will not work for this query.

For the RF 131 family, we used 7 query sequences against 6 targets, to get a total of 27 true hits. As seen in Table 5.4, most pair seeding methods achieve 27 true hits. Further, several of them – for example, the $NP(19.6)$ and $NP(19.1)$ methods – exceed the performance of the BLAST seed for false positives, as seen in Table 5.5. However, in this case, the *PH* method performs better than any pair seed. This family is a further illustration of the fact that while pair seeding methods can generally exceed *BLAST* performance, their thresholds have to be chosen carefully in order for them to do better than the *PH* method.

For the family RF 235, we used 9 query RNA sequences against 9 targets. This results in a maximum 45 true hits. This family offers an interesting example. Table 5.4 shows that the $NP(19.6)$ method fails to find even a single hit. Further, it does not find a single false positive either – see Table 5.5! This behavior can be explained by examining the contents of query sequences in more detail (shown in Appendix A). It turns out that all the members of the RF 235 family contain a large proportion (approximately 78%) of AU and UA pairs, at the expense of GC and CG pairs. However, if we derive the underlying pair probabilities from the Ribosum-95 scoring matrix shown in Table 2.2, we discover that the implied frequency of GC/CG pairs is much higher (approximately 40%). This

is reflected in the high scores for the alignments GC-GC and CG-CG, in comparison to those for AU-AU and UA-UA in Table 2.2. Due to this disagreement between the pair frequencies in the query and those implied by the scoring matrix, the threshold $T = 19.6$ proves to be too high for an alignment of two pair 4-mers where one of the 4-mers is from RF 235. Note that this does not indicate a failure of the pair seeding method. In fact, given an arbitrary query RNA, we can determine at runtime exactly how many 4-mers or 7-mers in it will score above the given T in a perfect match (as we have done in Appendix A). If the number is zero or close to zero, we can execute the search with a lower threshold score or execute a search based on nucleotide hashing. It is seen from Table 5.4 that executing the same 2-hit method with the lower $T = 19.1$ threshold results in all the true hits being found, with false positives comparable to the PatternHunter seed (Table 5.5), and much better than the *BLAST* method. In fact, the *PP(19.1)* method has the best performance for this family. This suggests that if the threshold is chosen carefully, it is possible to achieve *BLAST* sensitivity while significantly improving on *BLAST* specificity.

For the RF 265 family, three member sequences were identified as queries. We executed homology searches using all nine seeding methods described in Table 5.3, with every query against every target sequence – giving a total of eighteen search executions per method. The total number of true hits expected is six. Table 5.4 shows that all nine methods find 100% of the true hits. Further, it is seen that the *7mer(31.49)* method has the lowest false positives – 923nt, compared to the *BLAST* method's 11,774nt – see Table 5.5. While the *PH* method does better than *BLAST* with 5,403nt, it is surpassed by the *7mer(31.49)* approach. In fact, all three 1-hit Pair 7-mer methods and the *NP(19.6)* method do better than both *BLAST* and *PH* methods. These results support the conclusion that when the queries have long, high-scoring paired regions, the 1-hit 7-mer approach performs better than nucleotide seed based approaches.

Human Chromosome 22 Testing

Using the human chromosome 22 nucleotide sequence as a target, we executed the nine methods given in Table 5.3 with queries from each of the families RF 84, RF 103, RF 131, RF 235 and RF 265. For each family, we computed the total number of nucleotides marked as hits by each method – these are the "false positives" for that method. We then found the ratio of each method's value to that of the *BLAST* method. A ratio less

than 1.0 indicates that the method did better, while a ratio such as 1.5 indicates it did worse, than the *BLAST* one.

Table 5.6 below shows a summary of our testing. The false positive values in each column have been normalized using the value in the last row (value of the *BLAST* method). These values indicate that the *7mer(31.49)* method does consistently better than the *BLAST* over all families. Further, the *NP(19.1)* and *NP(19.6)* methods also do better than the *BLAST* method, assuming an appropriate threshold is chosen.

Table 5.6: False positives for the human chromosome 22 sequence. The false positives in each entry have been normalized using the value in the same column, in the *BLAST* row. Hence, a value below 1.0 indicates that the particular method has marked fewer false positives than the *BLAST* method in the Chromosome 22 target sequence.

Method	RF 84	RF 103	RF 131	RF 235	RF 265
7mer(29.86)	0.674	1.518	1.440	1.052	0.420
7mer(31.49)	0.151	0.235	0.061	0.131	0.067
7mer(32.79)	0.016	0.032	0.012	0.018	0.022
NP(19.6)	0.243	0.061	0.170	0	0.422
NP(19.1)	0.804	0.227	0.291	0.520	0.595
PP(19.6)	0.024	0.002	0.016	0	2.107
PP(19.1)	0.180	0.010	0.129	0.141	2.211
PH	0.248	0.379	0.306	0.438	0.311
BLAST	1.000	1.000	1.000	1.000	1.000

Testing with the chromosome 22 sequence indicated another useful feature of pair seeding homology methods. With a RF 84 query, the *7mer(31.49)* method marks 159,295 nucleotides as false hits. The *BLAST* method marks 703,044 nucleotides in the same target, using the same query, but only 2,704 of those nucleotides overlap with the ones marked by the *7mer(31.49)* method. In other words, 98.3% of the hits marked by the *7mer(31.49)* method are novel regions not identified by the *BLAST* method. This behavior is confirmed with other query and target sequences. We believe that our pair seeding methods complement traditional nucleotide seeding based methods, as they are able to identify potential homologous regions that the latter methods miss.

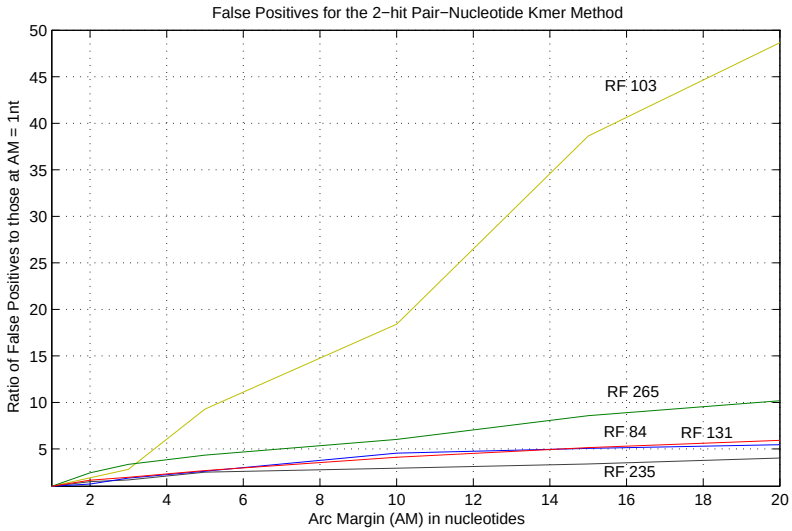

Figure 5.2: Variation of false positives with arc margin. We plot the arc margin (AM) along the x-axis. Along the y-axis is the ratio of false positives for a specific AM to those at AM = 1 nucleotide. A 2-hit method with pair 4-mers and nucleotide 8-mers was used.

Testing Variation in Arc Margins

Figure 5.2 below shows how the false positives vary with increasing arc margin, for the 5 RNA families we experimented with. We discern a general trend that false positives increase linearly with increasing arc margin values. This is as expected by the theoretical estimate of false positives varying as $\Theta((nwq)^2ar)$ for the method used. This indicates the importance of choosing a not too large, but large enough, value for the arc margin a. Note that this quantity is to be specified even with a 1-hit method.

5.5 Conclusions

We have conceptually developed and prototyped a set of RNA homology search methods based on the novel concept of hashing and aligning pair kmers. We have performed initial testing on member sequences of five RNA families. Further experiments with more RNA families would allow a comprehensive evaluation of these methods. Nevertheless, we note the following observations resulting from our testing.

First, for RNA families that have long, continuous paired regions, the 1-hit and 2-

hit pair seed methods achieve BLAST sensitivity while exceeding BLAST specificity. Secondly, for RNA families that have shorter or discontinuous paired regions, the 2-hit Pair-Nucleotide method can be used to achieve BLAST sensitivity and better-than-BLAST specificity, if the query contains enough high-scoring pair 4-mers.

Since query sequence and structure are precisely known prior to performing a target search, the query can be processed to extract useful information such as the number of pair 7-mers and pair 4-mers, how many of them would score above a given threshold if aligned to themselves, and the frequencies of pairs within the query. Appendix A contains such an analysis of all the queries used in our experiments. These quantities can then be used to make an advance decision regarding the usefulness of a given search method. If a given RNA query simply does not have enough paired content to justify a pair seeding approach, then we can execute a PatternHunter nucleotide search (with a seed of weight greater than 11), which still offers BLAST sensitivity with better-than-BLAST specificity. Note that such query analysis is unique to structured RNA molecules, and is not feasible for DNA and protein homology searches.

Hence, regardless of the RNA query provided, the results in this work suggest that pair seeding and spaced nucleotide seeds can be combined in a hybrid approach to offer a method that performs consistently better than a BLAST search. However, for such an approach multiple hash tables have to be built in memory for the given target, which imposes increased memory requirements.

5.6 Future Work

An immediate extension to this work would be to perform experiments on a range of additional RNA families. This would allow a comparison between pair hashing methods and traditional nucleotide hashing methods that is statistically of higher confidence.

Experiments with additional families could be useful in correlating query properties with predictions about a pair homology search method and threshold that would outperform BLAST and PatternHunter seeds. For example, Appendix A shows that queries in the RF 235 family have about 23 pair 4-mers, but none of them scores above $T = 19.6$ when aligned to itself. This tells us that a pair 4-mer based search with $T = 19.6$ will fail. However, it is harder to predict whether a particular method (and threshold score) will work *better* than any other method. For example, is it possible to state that "if a

query has 20 pair 4-mers, 12 of which score above $T = 19.6$ when aligned to themselves, then the 2-hit pair 4-mer method will (on average) outperform the 1-hit pair 7-mer, 2-hit Pair-Nt, PatternHunter and BLAST seed based methods"?

Secondly, the Ribosum scoring matrices were developed based on the SSU rRNA alignment of sequences from about 2,500 species. Specifically, background frequencies of nucleotides and pairs, and rates of substitution are as per the SSU rRNA family. From the experimental data presented above, it is evident that frequencies of pairs can differ significantly from one RNA family to another. Since such differences can be identified by analysing a given query, it would be useful to study methods of compensating for this discrepancy.

Lastly, as mentioned earlier, it would be useful to investigate more efficient algorithms and data structures to store pair of hits, for use in 2-hit search methods.

Bibliography

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[2] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[3] B. Brejová, D.G. Brown, and T. Vinar. Vector seeds: an extension to spaced seeds allows substantial improvements in sensitivity and specificity. In *Proceedings of the Third Workshop on Algorithms in Bioinformatics*, pages 39–54, 2003.

[4] B. Brejová, D.G. Brown, and T. Vinar. Optimal spaced seeds for homologous coding regions. *Journal of Bioinformatics and Computational Biology*, 1(4):595–610, 2004.

[5] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, pages 345—352. National Biomedical Research Foundation, 1978.

[6] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

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

[8] S.R. Eddy. Computational genomics of noncoding RNA genes. *Cell*, 109:137–140, 2002.

[9] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 2003.

[10] L.H. Hartwell, L. Hood, M.L. Goldberg, A.E. Reynolds, L.M. Silver, and R.C. Veres. *Genetics: From Genes to Genomes*. McGraw Hill, 2000.

[11] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[12] R.J. Klein and S.R. Eddy. RSEARCH: Finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4:44–59, 2003.

[13] B. Ma, J. Tromp, and M. Li. PatternHunter: Faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, 2002.

[14] L. Noe and G. Kucherov. Improved hit criteria for DNA local alignment. *BMC Bioinformatics*, 5(1):149–158, 2004.

[15] Z. Weinberg and W.L. Ruzzo. Faster genome annotation of non-coding RNA families without loss of accuracy. In *Proceedings of the Eighth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 243–251, 2004.

Appendix A

Structural Details of Queries

We present the results of analyzing all the query sequences that were used in our experiments. Each query's secondary structure was parsed to yield the following information:

Length. The query sequence's size in nucleotides.

#7-mers. This is the total number of paired regions of length 7 in the sequence. For example, the paired region AU-AU-GC-GC-UA-UA-CG-CG-AU-AU of length 10 has four 7-mers.

#Hashed 7-mers. A 7-mer is hashed if it scores above a threshold (*e.g.* 31.49) when aligned in a perfect match. The #hashed 7-mers in a query is instructive since any 7-mer that score below 31.49 in a perfect match is of little use in a search with $T = 31.49$.

#4-mers, #Hashed 4-mers These are analogous to 7-mer cases, for length-4 regions.

#4-mer Tuples. These are the #tuples (p_1, p_2) where p_1, p_2 are disjoint pair 4-mers.

#Hashed 4-mer Tuples. These are the #tuples (p_1, p_2) where both p_1, p_2 are hashed pair 4-mers for some threshold *e.g.* 19.6.

#Pair-Nt Tuples. These are the #tuples (n, p) where n is a nucleotide 8-mer, p is pair 4-mer and they are disjoint.

Table A.1: Secondary structure details of queries used in experiments - Part I.This table presents the queries from the RNA families RF 265, RF 235 and RF 131. Please see preceding text for an explanation of the fields.

Query	Size	#7-mers	#Hashed 7-mers	#4-mer	#Hashed 4-mers		#4-mer	#Hashed Tuples		#Pair-Nt	#Hashed Tuples	
			T=31.49		T=19.6	T=19.1	Tuples	T=19.6	T=19.1	Tuples	T=19.6	T=19.1
RF 265												
1	132	14	9	26	4	7	271	5	17	2695	419	733
2	132	14	9	26	4	7	271	5	17	2695	419	733
3	131	11	5	23	4	7	208	5	17	2364	415	726
RF 235												
1	131	13	9	26	0	6	274	0	9	2686	0	620
2	131	13	9	26	0	7	274	0	15	2686	0	722
3	128	9	6	22	0	6	192	0	9	2211	0	604
4	131	13	9	26	0	6	274	0	9	2686	0	620
5	131	13	9	26	0	6	274	0	9	2686	0	620
6	131	13	9	26	0	6	274	0	9	2686	0	620
7	131	13	9	26	0	6	274	0	9	2686	0	620
8	128	9	6	22	0	6	192	0	9	2211	0	604
9	128	9	6	22	0	6	192	0	9	2211	0	604
RF 131												
1	70	8	5	14	3	5	61	2	8	575	123	205
2	71	11	5	17	2	5	97	0	8	725	84	210
3	70	8	5	14	3	5	61	2	8	575	123	205
4	70	8	5	14	3	5	61	2	8	575	123	205
5	71	11	5	17	2	5	97	0	8	725	84	210
6	72	11	10	17	4	8	97	5	24	742	178	354
7	72	11	10	17	4	8	97	5	24	742	178	354

Table A.2: Secondary structure details of queries used in experiments - Part II. This table presents queries from the RNA families RF 103 and RF 84. Please see the preceding text for explanations of each field.

Query	Size	#7-mers	#Hashed 7-mers T=31.49	#4-mer	#Hashed 4-mers		#4-mer Tuples	#Hashed Tuples		#Pair-Nt Tuples	#Hashed Tuples	
					T=19.6	T=19.1		T=19.6	T=19.1		T=19.6	T=19.1
RF 103												
1	76	7	6	13	1	3	51	0	2	610	47	140
2	76	7	6	13	1	3	51	0	2	610	47	140
3	76	7	6	13	1	3	51	0	2	610	47	140
4	76	7	6	13	1	3	51	0	2	610	47	140
5	78	12	8	15	1	4	66	0	0	734	49	196
6	77	12	8	15	1	4	66	0	0	719	48	192
7	77	12	8	15	1	4	66	0	0	719	48	192
8	77	10	7	13	1	4	45	0	0	624	48	192
9	77	12	1	15	2	2	66	1	1	719	95	95
10	77	12	8	15	1	4	66	0	0	719	48	192
11	77	10	7	13	1	4	45	0	0	624	48	192
12	77	10	7	13	1	4	45	0	0	624	48	192
13	76	12	1	15	2	2	66	1	1	704	94	94
14	79	5	5	11	1	1	34	0	0	549	50	50
15	79	5	5	11	1	1	34	0	0	549	50	50
RF 84												
1	254	11	9	28	10	15	334	38	86	6320	2265	3395
2	254	11	9	28	10	15	334	38	86	6320	2265	3395
3	254	11	9	28	10	15	334	38	86	6320	2265	3395
4	254	11	9	28	10	15	334	38	86	6320	2265	3395
5	254	11	9	28	10	15	334	38	86	6320	2265	3395
6	253	11	9	24	6	11	235	13	44	5390	1353	2477
7	253	11	9	24	6	11	235	13	44	5390	1353	2477
8	253	11	9	24	6	11	235	13	44	5390	1353	2477
9	253	11	9	24	6	11	235	13	44	5390	1353	2477
10	254	11	9	28	10	15	334	38	86	6320	2265	3395
11	42	0	0	4	2	2	3	1	1	80	43	43
12	254	11	9	28	10	15	334	38	86	6320	2265	3395
13	255	4	2	10	2	4	30	1	5	2263	455	907
14	255	5	3	11	3	5	37	2	7	2489	681	1133

Appendix B

Consensus Structures of Query Families

We present below the consensus secondary structures of the five RNA families we used in our experiments. All the five figures have been reproduced from the RFAM [9] web site.

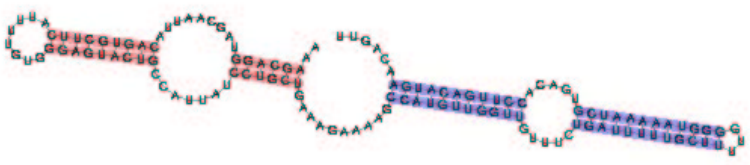

Figure B.1: Consensus secondary structure of the RF 265 family [9]

Figure B.2: Consensus secondary structure of the RF 84 family [9]

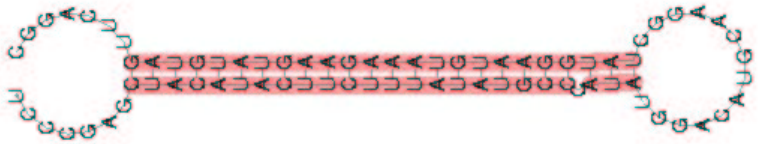

Figure B.3: Consensus secondary structure of the RF 103 family [9]

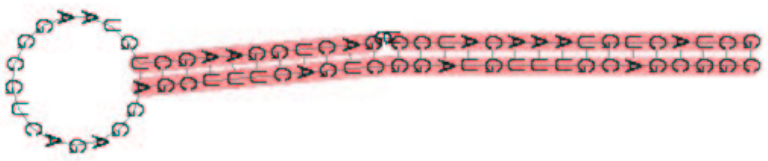

Figure B.4: Consensus secondary structure of the RF 131 family [9]

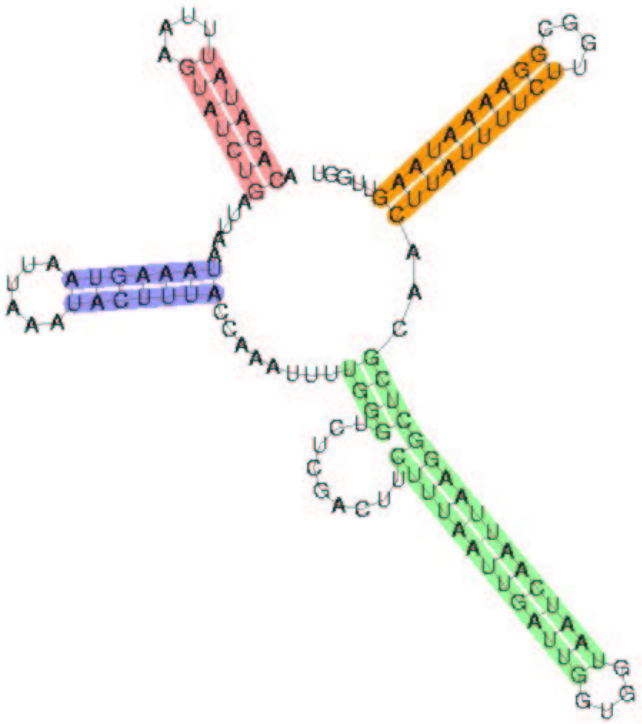

Figure B.5: Consensus secondary structure of the RF 235 family [9]