Efficient non-coding RNA gene searches through classical and evolutionary methods

Scott F. Smith

Electrical and Computer Engineering Department,
Boise State University, Boise, ID 83725-2075, USA
Fax: 208-426-2470 E-mail: sfsmith@boisestate.edu

Abstract: Successful non-coding RNA gene searching requires examination of long-range intramolecular base pairing possibilities. This results in search algorithms with extremely long run times such that large-scale use of the algorithms often becomes computationally infeasible. Methods for the efficient search of the solution space are examined. A review of the standard dynamic-programming covariance model search algorithm is given. An analysis of the statistically probable regions of the search space is undertaken and a method of limiting the traditional dynamic-programming algorithm to this region is shown. An alternative search method using a Genetic Algorithm (GA) which favours the probable region of the search space is also given.

Keywords: bioinformatics; covariance models; genetic algorithms; RNA database search.

Reference to this paper should be made as follows: Smith, S.F. (2009) 'Efficient non-coding RNA gene searches through classical and evolutionary methods', *Int. J. Computational Intelligence in Bioinformatics and Systems Biology*, Vol. 1, No. 1, pp.42–58.

Biographical notes: Smith is currently a member of the Computer Engineering Faculty within the Electrical and Computer Engineering Department at Boise State University where he does research on bioinformatics algorithms and hardware implementations. He is also a member of the IEEE Computational Intelligence Society's Bioinformatics and Biotechnology Technical Committee within which he has served as Proceedings Chair, Technical Co-chair and most recently General Chair for the committee's Symposium on Computational Intelligence in Bioinformatics and Computational Biology series. He holds a PhD in Electrical Engineering from the University of Idaho and another PhD in Economics from the State University of New York.

1 Introduction

MOST of the literature on gene finding is focused on the search for protein-coding genes. These are portions of an organism's genomic DNA sequence which specify the order of amino acids in the organism's proteins. The DNA sequence is *transcribed* into messenger RNA (mRNA) which is later *translated* into protein at a ribosome. The translation uses the genetic code to convert three-nucleotide codons into amino acids. The purpose of the mRNA molecule is to transport information about the eventual amino

acid sequence from the chromosome storing the information to the ribosome producing the protein.

There are many other types of RNA molecules which are also transcribed from the chromosomal DNA which do not carry protein coding information. These include transfer RNA (tRNA), ribosomal RNA (rRNA), RNA components of mixtures of protein and RNA called ribo-nucleo-proteins (RNPs), and many others (Gesteland et al., 2006). These RNA molecules which do not carry amino acid sequence information are known collectively as *non-coding RNA* (ncRNA) or *functional RNA*. The function of these RNA molecules depends directly on interaction of the molecule with other molecules in the cell (rather than indirectly through a protein product as is the case with mRNA). This direct interaction makes the shape of the ncRNA molecule important. The single most important determinant of the ncRNA molecule shape is its intermolecular base pairing pattern, also known as its *secondary structure*. The *primary structure* of the ncRNA molecule is the sequence of nucleotides (A, C, G, or U) along the molecular chain.

The likelihood that a particular position in the ncRNA molecule will attach itself to another particular position in the same molecule (that the positions will base pair) depends on the nucleotides at the two positions. A (adenine) prefers to base pair with U (uracil) and G (guanine) prefers to base pair with C (cytosine), although the G-U base pair is also not uncommon. In all, there are 16 possible pairings which may have different likelihoods of forming a base pair either generically or when viewed from specific locations within the nucleotide chains of given ncRNA families. In the type of model used to search for genes of members of an ncRNA family in this paper, the data from known family members is used to estimate the probabilities of the 16 possible parings at individual positions.

The fact that interactions between nucleotides of the ncRNA molecule that may be very far apart in the molecular sequence (but very near in physical proximity) are important makes searching for ncRNA genes more difficult. When searching for protein-coding genes, heavy use can be made of the tendency to primary structure conservation. There are often regions of the amino acid sequence that can not radically change without destroying the function of the protein. This is not the case with ncRNA, where compensating variations in nucleotides that do not interfere with base pairing are quite frequent. For example, two positions in the ncRNA sequence which are very distant in sequence position might be A–U in one family member and C-G in another family member. The simultaneous mutation of A to C and of U to G have maintained the ability of the two positions to base pair. The models that allow the capture of primary structure for protein homology search (such as profile hidden Markov models) are not sufficiently powerful enough to capture the necessary secondary structure information needed for ncRNA gene search.

The profile Hidden Markov Model (HMM) (Eddy, 1998) which may be used as part of protein gene search is an example of a *regular grammar* in the Chomsky (Chomsky, 1959) hierarchy of transformational grammars. In order to model base pairing in a similar framework as an HMM, it is necessary to move up one level of the hierarchy to the *context-free grammar*. The other two even higher levels of the hierarchy, the *context-sensitive* and the *unrestricted grammar*, are rarely seen in biological sequence analysis. A common probabilistic model using the context-free grammar for ncRNA gene search is the *covariance model* (Eddy and Durbin, 1994) which has much the same flavour as the HMM. The covariance model will form the basis for the rest of this paper.

Efficient search for ncRNA genes will be presented as either a limitation of the search space in conjunction with the classical (dynamic programming) method of searching with the model or as a Genetic Algorithms (GA) search based on the same covariance model as the classical search method.

The contention that the classical covariance model search is inefficient comes from observation of the actual distribution of ncRNA family members within the search space used by this method. The classical method requires the user to specify the maximum sequence length D of new members of the family which can be found. The processing time required for the search is proportional to $aD + bD^2$, where the weights a and b depend on the particular model. The value D must be chosen large enough such that true family members with large net numbers of insertions are not missed, yet small enough to keep computation time reasonable. A major problem is that the parameter D is used for evaluation of every portion of the model. It will be shown that this leads to large parts of the search space being completely unrealistic. In addition, both the proposed limited classical search and the genetic algorithm search do not require the user to specify the parameter D. In the classical case, the data used to build the model guides the choice and in the genetic algorithm case, the fitness landscape itself determines where to search.

The following section gives an overview of covariance models and the traditional use of dynamic programming for database search. Section 3 investigates the statistics of sequence length in different parts of the model. A simple redesign of the classical covariance model search algorithm that takes advantage of the search statistics to limit the search space is discussed in Section 4. The alternative GA search method is presented in Section 5. Finally, Section 6 gives some concluding remarks.

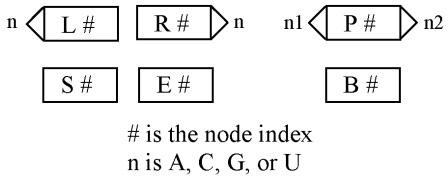
2 Dynamic programming for non-coding RNA gene search with covariance models

Covariance models (Durbin et al., 1998) are composed of *nodes* which can be arranged in the form of a binary tree. The root of the tree (the root start node) is normally shown at the top. There are six types of nodes, three of which are symbol *emitting* and three are *silent*. The silent nodes are used to give structure to the tree and place the emitting nodes in the proper orientation with respect to each other. The silent node types are *end* (E), *bifurcation* (B), and *start* (S). Each branch of the tree has an S node at the top and either a B or an E node at the bottom. B nodes have exactly two S nodes as children and E nodes have no children. S nodes come in three flavours, the root S node (of which there is only one), left child S nodes, and right child S nodes. The internal structure of the three S node flavours varies slightly as described below.

All nodes between the S node at the top and the B or E node at the bottom of each branch are symbol emitting nodes. The emitting nodes are of three possible types, *pair-emitting nodes* (P), *left-emitting nodes* (L), or *right-emitting nodes* (R). The L and the R nodes are known as *single-emission nodes* since they emit one symbol while the P node emits two. Emitting nodes add to the subtree below them by placing a new symbol (nucleotide) on either the left (L node), right (R node), or both (P node) ends. The E node is a maker for the null sequence and gives a location to start building upon. Every node is given an index number with the root S node indexed at 0. For all nodes other than B nodes, a child node always has and index one greater than that of the parent. For B nodes the left child always has an index one greater than the B node. The right B-node child has

an index that results from a depth-first assignment of node numbers where the left path is always followed first. Figure 1 shows the icons that will be used for the different node types in the tree. In addition to the node index #, symbol-emitting nodes are labelled with the symbol with highest probability of emission (L node and R node) or symbol pair with highest probability of emission (P node), even though any symbol may be emitted.

Figure 1 Covariance model node icons



The arrangement of the covariance model tree can be determined from a multiple alignment of the member sequences from which the model is to be estimated. The *consensus* columns of the multiple alignment are those which are 'normally' present. Sequences that have a nucleotide in a non-consensus column are assumed to have it due to an 'unusual' insertion in the sequence with respect to the family. Sequences that do not have a nucleotide in a consensus column are assumed to not have it due to an 'unusual' deletion of that position relative to the family. The definitions of 'normally' and 'unusual are not precise, but a common way of defining a column as a consensus column is if one-half or more of the sequences in the multiple alignment have a nucleotide in that column.

The generation of a multiple alignment from groups of sequences is outside the scope of this paper, but good discussions on multiple alignment can be found in Mount (2001) and Eidhammer et al. (2004). Loosely speaking, nucleotide positions in the same column of different sequences come from related parts of the two sequences with possible insertions and/or deletions in either sequence with respect to a common ancestral sequence.

In order to form a covariance model, it is also necessary that the multiple alignment be annotated with secondary structure. Each column should be listed as either non-consensus, unpaired consensus, paired columns also need notation as to which other column they base pair with. If one makes the assumption that there are no pseudoknots, then a simple notation using the symbols '.', '–', '<', and '>' in each column completely specifies the secondary structure. Pseudoknots are a structural feature that are important in a few ncRNA families, but which are absent in most. The standard covariance model can not represent pseudoknots, so the simple secondary structure notation used here is not limiting. If the ncRNA family actually contains a pseudoknot, then a somewhat less powerful, but usually still useful model can be formed with no pseudoknot simply by ignoring some of the true base pairings and treating some of the multiple alignment columns as if they were unpaired.

Figure 2 shows an example of a multiple alignment with secondary structure annotation. This alignment will be used to construct an example covariance model tree in the text that follows. The example is the alignment of the 14 sequences in the U12 ncRNA family from the Rfam database (Griffiths-Jones et al.,2005). The alignment is broken into two halves in order to fit on the page with the left half at the top of the figure and the right half at the bottom. The last two lines of the alignment are the

consensus sequence and the secondary structure respectively. When the secondary structure has a '.' symbol, it is a non-conserved column and does not appear as a node in the model. A '–' means the column is single emission and will end up being an L or an R node. A '<' and matching '>' represent a P node, where matching is the normal nesting used for parenthesis. The consensus sequence shows the most probable symbol to be emitted (L and R nodes) or most probable pair of symbols (P node). The first five '<' symbols and the first five '>' form a group know as a *stem* (labeled 'stem 1' in the figure). The P nodes of a stem will appear together in the same branch of the covariance model tree. Stems can appear sequentially in the alignment or they may be nested inside other stems. This pattern of sequential vs. nested stems is what determines the organisation of the binary tree for the covariance model. The single emitting nodes between the first five '<' and first five '>' (marked with a '–') are known as a *hairpin loop* and hairpin loops can always be represented by a series of L nodes. They could also be represented by many different mixes of L and R nodes, but covariance models always default to using L nodes when either L or R is possible in order to make the models unique representations of a given multiple alignment. It is not always possible to use an L node for single emission and therefore R nodes are available.

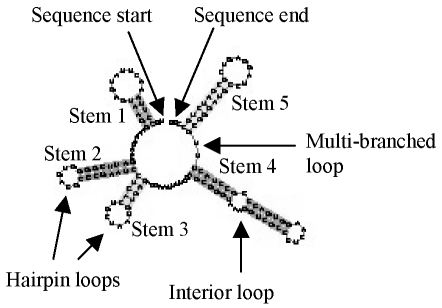
Figure 2 Example multiple alignment with secondary structure annotation (U12 family)

```
 1  UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACAACU......CGGGGUGACGCCCGAGUCCUCACUACUGAUGUGAGA
 2  AGCCUCAAA....CUUAAGGG.UAAGGAAAAUAAUGAUU......GCAGCAG.UGCCUGAGUCCUG.CUGCUAAUG..AGA
 3  UGCCUUAAA....AUUAUAAG.UAAGGAAAAUAAUGAUU......CGGGGUGACGCCCGAGUCCUCACUGCUUAUGUGAGA
 4  UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAGUCCUCACUGCUUAUGUGAGA
 5  UGACUUA.A....CUU...AGCUAAGGAAAAUAAUGGUU......CAGGGUGACAUCAGAGUCCUUGUUGCUUCUAUGAGA
 6  UGCCUUAAA....AUUAUAAG.UAAGGAAAAUAAUGAUU......CGGGGUGACGCCCGAGUCCUCACUGCUUAUGUGAGA
 7  GGUAUUAAGAGUUCUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAGUCCUCACUGCUUAUGUGAGA
 8  UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACACCCGAGUCCUCACUGCUUAUGUGAGA
 9  UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAGUCCUCACUGCUUAUGUGAGA
10 UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGA
11 UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGA
12 UGUCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUUGUUAUUCGGGGUGAUGCCCGAAUCCUCACUGCUAAUGUGAGA
13 UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGA
14 UGUCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUUGUUAUUCGGGGUGAUGCCCGAAUCCUCACUGCUAAUGUGAGA

   UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU......CGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGA
   --<<<<<--....---------.>>>>>---------<<<<......<<<<------>>>>>>>>>><<<<<-------->>>>>-
    stem 1              stem 1           stem 2          stem 2   stem 3      stem 3

 1  GGAAUUUUUGUGCGGGUACAGGUCGUCCCC.GG.GUGACCCGCUUACUUCGCGGGAUGCCCAGGUGCAAUGAUCUGCCCG
 2  UGAAUUUUUGAGUGGGUAAAGGAUGUCCCUAAA.GCAGCCCACCUACA..AAUAUAUAAAUAUGAGAAGAGGGCUGGGGG
 3  AGAAUUUUUGAGCGGGUAUAGGUUGCAAUCUGAGGCGACCCACCUACUUUGCGGGAUGCCUGGGUAACGCGAUCUGCCAG
 4  AGAAUUUUUGAGCGGGUAUAGGUUGCAAUCUGAGGCGACCCGCCUACUUUGCGGGAUGCCUGGGUGACGCGAUCUGCCCG
 5  ..AUUUUCUGAUCUGGUAAAGGUGGCCCCAAAG.GAGACCAGUGUACAAUGCA.GAUGCUUACAUAUAGAAACAUGCCAG
 6  AGAAUUUUUGAGCGGGUAUAGGUUGCAAUCUGAGGCGACCCACCUACUUUGCGGGAUGCCUGGGUAACGCGAUCUGCCAG
 7  CGAAAUUUUGAGCGGGUAUAGGUUGCAAUCUGAGGCGACCCGCCUACUUUGCGGGAUGCCUGGGUAACGGGAUCUACCCG
 8  CGAAUUUUUGAGCAGGUAUAGGUUGCAGUCUGAGGCGACCCGCCUACUUUGUGGGAUGCCUGGGUAACUCGAUCUGCCCG
 9  AGAAUUUUUGAGCGGGUAUAGGUUGCAAUCUGA.GCGACCCGCCUACUUUGCGGGAUGCCUGGGUGACGCGAUCUGCCCG
10 CGAAUUUUUGAGCGGGUAAAGGUCGCCCUCAAG.GUGACCCGCCUACUUUGCGGGAUGCCUGGGAGUUGCGAUCUGCCCG
11 CGAAUUUUUGAGCGGGUAAAGGUCGCCCUCAAG.GUGACCCGCCUACUUUGCGGGAUGCCUGGGAGUUGCGAUCUGCCCG
12 CGAAUUUUUGAGCUGGUAAAGGUCGCCCCUAAG.GUGACCAGCCUACUUUGCGGGAUGCCUAGGAGUCGCGAUCUGCCUG
13 CGAAUUUUUGAGCGGGUAAAGGUCGCCCUCAAG.GUGACCCGCCUACUUUGCGGGAUGCCUGGGAGUUGCGAUCUGCCCG
14 CGAAUUUUUGAGCUGGUAAAGGUCGCCCCUAAG.GUGACCAGCCUACUUUGCGGGAUGCCUAGGAGUCGCGAUCUGCCUG

   CGAAUUUUUGAGCGGGUAAAGGUCGCCCCCAGG.GUGACCCGCCUACUUUGCGGGAUGCCUAGGUGCCGCGAUCUGCCCG
   ----------<<<<<<<<<<-<<<<<<<<<<->.>>>>>>>->>>>>>>>--<<<<<<<<<<---------->>>>>>>>>>---
                stem 4                stem4          stem5           stem 5
```

The reason for calling the five-base-pair group a stem in the text above can be seen more clearly with the help of the secondary structure diagram in Figure 3. The base pairs are shaded grey in this diagram. Even though stem 4 is interrupted with an *interior loop*, this does not cause branching and will therefore be treated as a single stem for the purposes of this paper. Since there are five hairpin loops, one at the end of each stem, there will be five E nodes in the model tree. Immediately above the E nodes will be a series of L nodes representing the nucleotides of the hairpin loop and above the L nodes will be a series

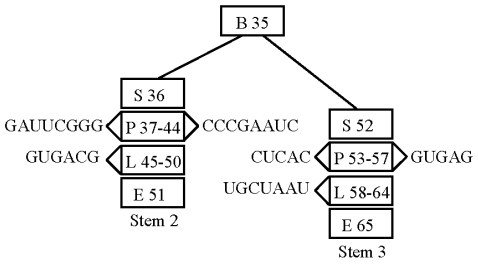
of P nodes representing the stem. In stem 4, the series of P nodes will be interrupted with L and R nodes to represent the interior loop.

Figure 3 Example secondary structure diagram (U12 family)

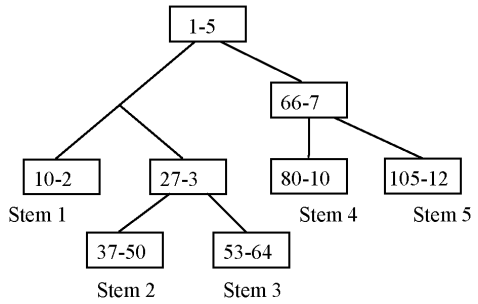


The resulting covariance model tree is shown in Figure 4. In Figure 4a only the portion of the tree associated with stem 2 and stem 3 is shown. Multiple consecutive nodes of the same type have been collapsed into a single node and labelled with a range of node numbers. For example, Figure 4(a) shows a single node labelled 'L 45–50' which is really six L nodes in the true covariance model tree in a vertical line with L 45 on top and L 50 on the bottom. Figure 4(b) shows an even more condensed version of the overall model tree. In this figure, the node number ranges in the boxes list the indices of the symbol emitting nodes in the branch (L, R, or P) with no indication of which types of emitting nodes are in the branch.

Figure 4 Example covariance model tree: (a) fragment representing stems 2 and 3 and (b) condensed representation of full tree



(a)

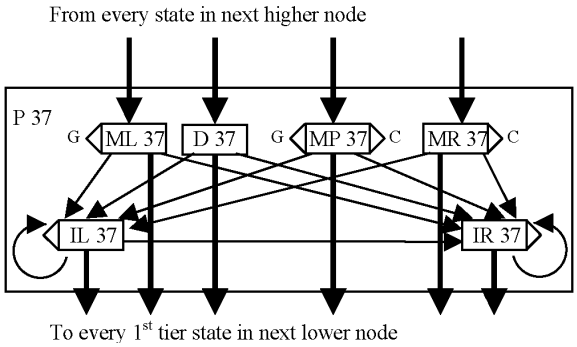


(b)

The covariance model tree only shows the consensus secondary structure and consensus emission symbols. However, real ncRNA genes have insertions and deletions with respect to the consensus sequence. In order to make insertions and deletions, the model nodes have an internal *state* structure with between one and six internal states, depending on the node type. There are four types of silent states: start (S), end (E), bifurcation (B), and delete (D). There are three types of symbol-emitting states: left (L), right (R), and pair (P). Some nodes have two symbol-emitting states of the same type, so the role of the state is usually further denoted by adding a M (for match) and I (for insert) before the state type, so these states are given as IL, IR, ML, MR, or MP (there are no IP states). Whether the role of a state is to match or to insert, its implementation is identical. All of the connections between states have *state transition probabilities* and all of the symbol-emitting states have *emission probabilities* for each possible symbol or pair of symbols. These probabilities are estimated from the observed frequency of occurrence in the multiple alignment plus pseudocounts to avoid zero probabilities.

The node type with the most complicated internal state structure, the P-type node, is shown in Figure 5. Specifically, Figure 5 shows the internal structure of a P-type node labelled with the emissions associated with node P 37 in stem 2 of the U12 example above. The portion of the state structure associated with the consensus is the MP state, which emits a G-C pair with highest probability. If a database sequence does not contain this pair at all, then the D state will be visited. If only the right half of the pair is in the database sequence, then the MR state is visited and similarly the ML state is visited if only the left half of the pair is present in the database. The IL and IR states allow any number of additional symbols to be emitted before the consensus pair is emitted. The fat arrows in the figure indicate multiple state transitions into or out of the node. State transition probabilities for the path through the MP state only should be the highest. The degree to which other transition probabilities are lower than the consensus path depends on the data in the multiple alignment.

Figure 5 Internal state structure of a P-type node



Once a covariance model has been estimated, it may be used for database search. Estimation of the model is a rather trivial computational burden, but database search with the model requires computational resources that are often more than what is available. The classical way to do this database search is with the CYK algorithm, which is based on dynamic programming. The algorithm progresses through the database sequence one sequence position at a time. Each sequence position is taken as a potential ending position of a gene in the ncRNA family of the covariance model. At each ending position,

the model is evaluated for every possible sequence length extending up to that ending position in the range 0 to some user-specified maximum D. The parameter D is chosen to be larger than the consensus sequence length. The model tree is evaluated for every database ending position j and for every sequence length d in [0, D] by processing the states starting at the end states and working up toward the root start state.

The overall score for each ending position is the best score taken over all sequence lengths d for the root start state. The individual state transition probabilities and emission probabilities are converted to log likelihood ratios before the search algorithm is run, so scoring involves summation and maximum operations and the result can be interpreted as a log likelihood ratio. More formally, the CYK algorithm is:

for j = 0 to L, d = 0 to min(D, j), v = M to 0 {

case type(v) is

E: if d = 0 then s(j,0,v) = 0; else s(j,d,v) = –Infinity

D or S: s(j,d,v) = max(over children c)[s(j,d,c) + trans(c,v)]

L: if d = 0 then s(j,0,v) = –Infinity; else s(j,d,v) =

emit(l,v) + max(over children c) [s(j,d 1,c) + trans(c,v)]

R: if d = 0 then s(j,0,v) = –Infinity; else s(j,d,v) = emit(r,v) + max(over children c) [s(j – 1,d – 1,c) + trans(c,v)]

P: if d < 2 then s(j,d,v) = –Infinity; else s(j,d,v) =

emit(l,r,v) + max(over children c) [s(j-1,d-2,c) + trans(c,v)]

B: s(j,d,v) = max(over k in 0 to d) [s(j-k,d-k,lc) + s(j,k,rc)]

}

The algorithm loops over all ending positions j, sequence lengths d, and state indices v to evaluate the score $s(j, d, v)$, where the length of the database sequence is L, the maximum search length is D, and the number of states is $M + 1$. E states set the score to 0 if the evaluation sequence is null and $-\infty$ if not (since a non-null sequence is not possible at an end state). D and S states are treated identically and simply find the best child score plus transition score to the state over all immediate children of the state c. L states seek to add a symbol to the left end of the sequence passed up from each child, so the appropriate child scores are for sequences of length one less ($d - 1$), but with the same end position j. Since an L state adds a symbol, the resulting sequence must be at least of length 1, so the score is set to $-\infty$ for $d = 0$. R states add a symbol to the right, so the appropriate child state scores are for length one less ($d - 1$) and end position one less ($j - 1$). The result of an R state must be a sequence of length at least one, so the score is set to $-\infty$ for $d = 0$. P states add a symbol to each end, so child scores are for length two less ($d - 2$) and end position one less ($j - 1$). P states must result in a sequence of length at least two, so the score is set to $-\infty$ for $d < 2$. The B state concatenates two sequences such that the overall sequence length is d and ends at j and the score is maximised. All possible right sequence lengths k are tried, where the left sequence must be of length $d - k$ to make the overall sequence length d.

The CYK algorithm uses $-\infty$ as a marker for impossible situations (actually, the largest negative value that the processor can represent is normally used) and this value never shows up as the solution since it always enters into a maximum calculation with some possible situation which has a log likelihood ratio greater than $-\infty$ (guaranteed since pseudocounts eliminate any probabilities exactly equal to 0). It would be possible to also use $-\infty$ as an estimated score for any very improbable situation rather than actually calculating the large negative score. This $-\infty$ will also not show up as the final answer since some more probable path through the model will take its place. It is only necessary that the situation be improbable enough that it does not actually appear as a true family member that one seeks in the database, in which case the family member will not be found. This is precisely what is done in what follows when the search space is limited for the classical algorithm and a child score is not available since it was deemed too improbable to bother calculating.

3 Probable regions of the gene search space

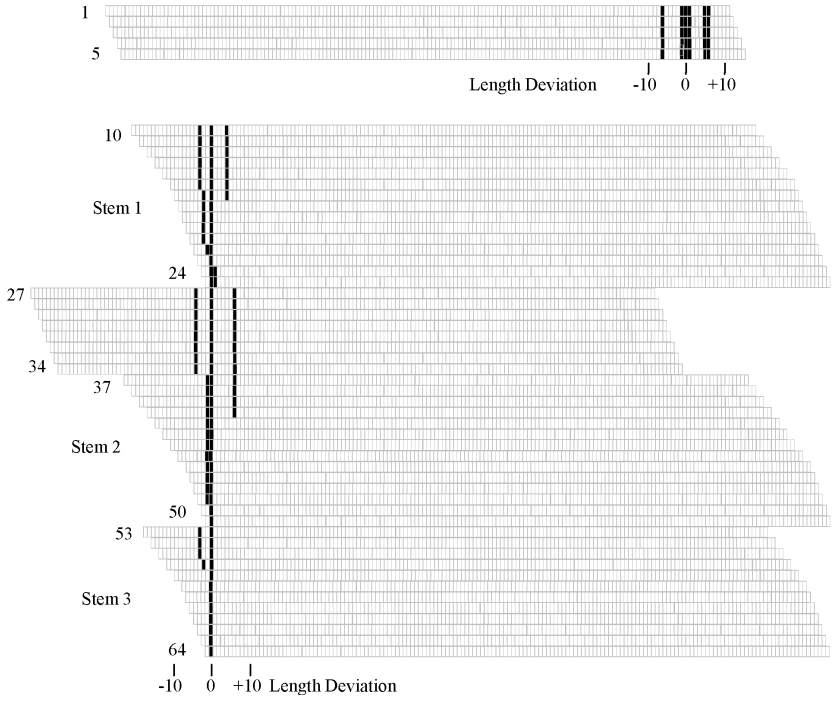
The standard way to search for non-coding RNA genes using a covariance model is to investigate every possible end position in each of the database sequences and every possible sequence length between 0 and a maximum length D. This is done by calculating the best score for every end position and sequence length at every state in the model. Dynamic programming is employed such that the best score for a position, length, and state is found as the best among increments to the best scores of a state's children. This is possible since the best scores at the child nodes are independent of the increments (transition and emission probabilities). If dynamic programming could not be used, then the calculation would be infeasible for any realistic model and database sizes. Even with dynamic programming, the computational burden of database search with covariance models is still extremely large.

This section investigates the portion of the search space in the sequence length dimension that is observed to contain solutions for ncRNA families in practice. The focus will be at the node rather than state level, since this is the level at which there is a one-to-one correspondence with multiple alignments. To use this information at the state level, one assumes that all states inside a node experience the same distribution of sequence lengths associated with best scores as the overall node. This is exactly true of all states except the insert states. With the exception of very long observed insertions, the assumption is also very good for the insert states. To guard against problems with insert states and long insertions, a superset of observed lengths from the insert state's own node and the node of its children (all the insert state's children are always in the same node) can be used.

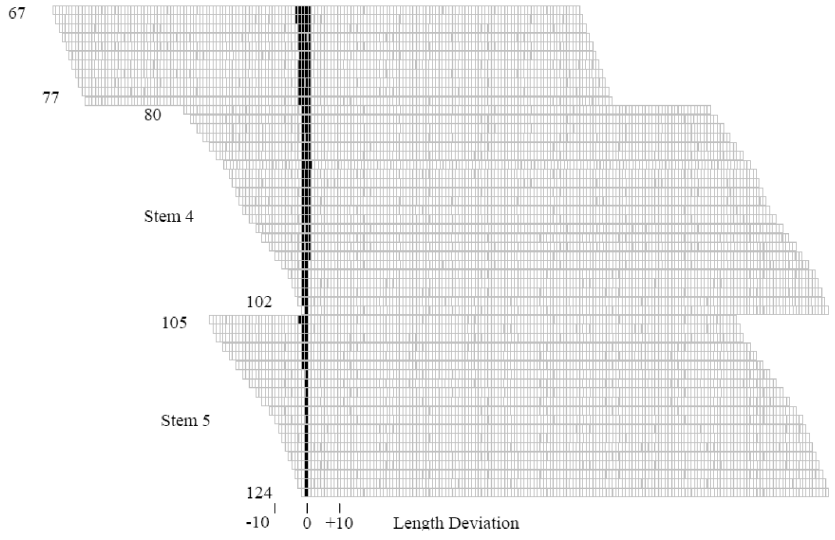
Figure 6(a) (parts a and b) show the portion of the sequence length dimension of the search space which is used by solutions that lead to the best score for database positions that are associated with true family members of the U12 ncRNA family. Only symbol-emitting nodes are shown, but silent node statistics can be inferred from those of the silent node's parents and children. Each node's row of the figure has 161 boxes associated with the possible sequence lengths in the range 0 to 160 (D was chosen to be 160 when the Rfam database was generated). The inside of the box is black if any family member uses that length in the given node to generate the best score. If the best scores of all the remaining boxes had been set to $-\infty$ instead of being

calculated, the overall scores for every true positive position would remain unchanged and the score for every false position would be less than or equal to the score of the standard method.

Figure 6 Search space usage: (a) U12 nodes 1 – 64 and (b) U12 nodes 67 – 124



(a)

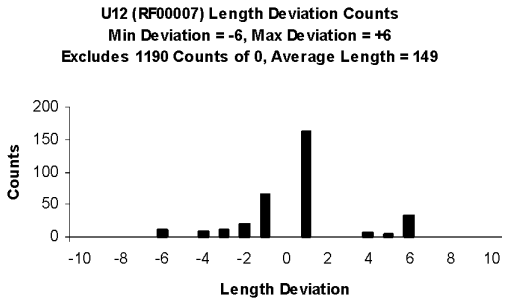


(b)

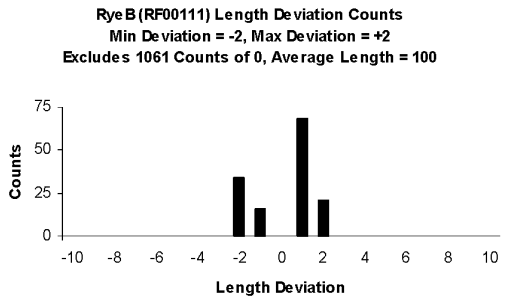
The search space in Figure 6(b) has been arranged with *length deviation* rather than sequence length on the *x*–axis in order to make the clustering of the true solutions more clear. Length deviation is defined here as the sequence length of the true solution at a node minus the consensus sequence length at a node. The consensus sequence length at a node is found from that portion of the consensus sequence which is represented by the subtree for which the node is the root. Similarly, the solution length at a node is the portion of the database sequence which has generated the best score at the node which will form a portion of the best score for the overall model. Nonzero length deviations are the result of insertions net of deletions in the partial database sequence fitted to the model subtree rooted at the node.

In the U12 example shown, there is never a length deviation greater than 6 in magnitude even though the standard solution method searches deviations as large as 159. This clustering of solutions very close to zero length deviation is not at all unique to U12, but rather quite widespread in the Rfam database families. Figure 7 shows histograms of length deviations for all emitting nodes in four ncRNA families. Even though length deviations in the traditional search space extend well beyond those shown in the figure, there is no situation in any of the four cases where those deviations were associated with the best score in practice. In the histograms, the counts for 0 length deviation have been omitted since they make the other counts hard to read. The average sequence lengths of 149 (U12), 61 (miR-9), 100 (RyeB), and 94 (snoR41) are a good approximation to the consensus sequence length and the maximum search length D would be chosen greater than these values. In all cases the observed length deviations are tightly packed about 0 deviation. This fact will be used to motivate the search space limitations for the classical and the genetic search algorithms discussed in the following two sections.

Figure 7 Length deviation histogram: (a) U12; (b) RyeB; (c) miR-9 and (d) snoR41

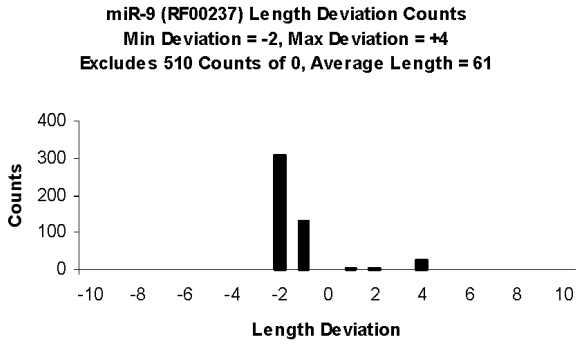


(a)

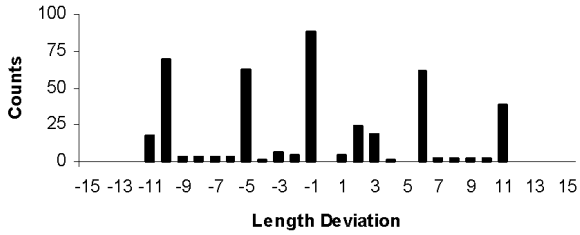


(b)

Figure 7 Length deviation histogram: (a) U12; (b) RyeB; (c) miR-9 and (d) snoR41 (continued)



(c)



(d)

4 Limiting the search space for the classical algorithm

The classical method for searching nucleotide databases for ncRNA genes using covariance models is to employ dynamic programming and to search sequence lengths in the range [0, *D*] at every state. This method already limits the search space to genes with overall length no more than *D*. The maximum search length parameter *D* is chosen based on the observed lengths of known genes and there is no guarantee that unknown genes in the family in fact have length no more than *D*. The investigation of the previous section indicates that this choice of a uniform sequence length search range for all states is probably not very efficient. Clearly, it is highly unlikely that states very close to the end states will have best-scoring lengths close to the maximum expected overall length of the gene when a true family member is found. Likewise, states very close to the root start state are not likely to have best-scoring lengths close to zero.

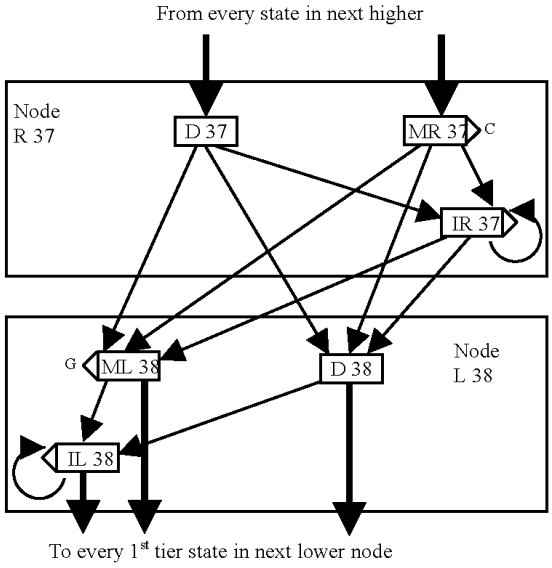
There are several ways to choose the range of the sequence length dimension of the search space more efficiently. The simplest is to specify a maximum expected sequence length deviation rather than a maximum expected overall sequence length as is currently done. Such a parameter could still be dependent on the ncRNA family (as *D* currently is) and might be chosen simply as twice the maximum of total insertions and of total deletions observed in any known family sequence. A more refined method could compile statistics on length deviations at each node in each model. From this a list of maximum and minimum lengths for the search space at each node can be compiled and read into the search program in addition to the covariance model parameter data. The non-uniform search of sequence lengths at different nodes may result in some of the

child scores needed by a parent node to not have been computed by the child node. This can be easily handled by initialising all scores to $-\infty$ before calculation begins.

One minor difficulty with using node-dependent minimum search lengths $DN_{\min}(n)$ and maximum search lengths $DN_{\max}(n)$ at each node n, is converting these to state-dependent minimum search lengths $DS_{min}(v)$ and maximum search lengths $DS_{\max}(v)$ at each state v. This is simple in the case of all non-insert states, where the appropriate values of $DS_{\min}(v)$ and $DS_{\max}(v)$ are simply the node-dependent values for the node in which the state is contained. This is because the non-insert states always generate a score which is used by a parent in the next higher node. This is exactly the place for which the node length deviation statistics are generated.

As shown in Figure 8, insert states are somewhat more complicated. Insert states are the only states that every have self loops and insert states are a symbol-emitting type of state. Furthermore, insert states are the only type of state that ever generates a score that is consumed by a parent located in the same node. As a result, using the node minimum search length for the insert state may result in the insert state not generating scores for short enough sequence lengths. It is recommend here that the maximum search length $DS_{\max}(v)$ for an insert state v equal the maximum search length $DN_{\max}(n)$ of the node n that the insert state is in. The minimum search length $DS_{\min}(v)$ of the state should equal the minimum search length $DN_{\min}(n + 1)$ of the child node $n + 1$ of the node n that the insert state is in. Since bifurcation nodes do not contain insert states, there is always one child node and its index is always one greater than the index of the parent node.

Figure 8 Insert state search length determination



5 Limiting the search space using a genetic algorithm

So far, limiting the search space has only been discussed in terms of reducing the number of sequence lengths evaluated at each node. It has still been assumed that a complete search will be made at each possible nucleotide sequence end position in the database

sequence. This can be seen as possibly inefficient since evaluation of a subset of sequence lengths at a given database position may give one information about the likelihood that evaluation of the full set of sequence lengths will return a high score. Given limited computing resources, it may be better to concentrate the search on positions in the database that produce good scores on a small subset of the sequence lengths at each node. A GAs approach to this idea is presented in this section.

Fitting a database sequence to a covariance model is equivalent to choosing a pattern of insertions and deletions with respect to the consensus sequence represented by the model. In order to easily apply GAs (and many other search methods) to this search it would be helpful to have a fixed-length representation of the pattern of insertions and deletions (Yadgari et al., 2001). A fixed-length vector of non-negative integers of length equal to that of the consensus sequence can be made to do the job. A value of zero at a vector position implies that the associated consensus position does not exist in the database sequence. A value of one indicates that the database does have a nucleotide at the consensus position and that no nucleotides have been inserted to the right of this position in the database sequence. A value m greater than one in the vector represents $m - 1$ inserted nucleotides in the database to the right of the consensus position. Using this notation the vector of all ones implies no insertions or deletions and is equivalent to a length deviation of zero at all nodes of the covariance model.

The initial generation of the genetic algorithm should always include the vector of all ones and other individuals in the initial generation should mostly have small length deviations at each node such that the search is initially concentrated at solutions with high probability of being near a true solution. This can be accomplished by introducing a small number of mutations of small magnitude relative to the vector of all ones. To match mutations in the GA to those observed in nature, it is desirable that insertions and deletions appear in groups rather than independently scattered across the sequence. This is automatically accomplished by the representation for insertions if we make changes in individual integers in small amounts, but sometimes in amounts greater than one. To accomplish this grouping for deletions, it is necessary to select a range of positions in the vector to set to zero all at once.

A major advantage to using GA over other search methods is the ability of the GA to use crossover. A good pattern of insertions and deletions in one portion of the vector will lead to a high contribution to the score. A poor pattern in another portion of the vector will lead to a score contribution that is on average about the same as fitting a random non-family-member location in the database. The sum of the two contributions will generally be way above the background noise of the scores. This will tend to cause two solutions with different good pattern regions to both remain in the population as long as steps are taken not to converge too quickly and maintain population diversity. Given enough crossover attempts, the two partial good patterns should eventually merge and create a solution with a higher score than either of the two partial good solutions.

Another advantage of GA is that they incorporate randomisation. Improvement in the score of a solution often requires the simultaneous insertion of a few nucleotides in one place and deletion of a few in another place. This has the effect of shifting a range of nucleotides in the middle of the sequence while leaving both ends in place. Doing either the insertion first or the deletion first often results in a decrease in score. Gradient assent algorithms can not escape the local minimum caused by the need to do simultaneous changes. Figure 9 shows an example of this using one of the U12 sequences of the multiple alignment of Figure 2 (sequence number 2) and the consensus sequence from the

same figure. The first pair of sequences shows the correct alignment of the database sequence with the model consensus sequence as defined by the original multiple alignment with any non-consensus columns in common (dots in both) removed. The underlined nucleotides throughout the figure indicate positions that are correctly aligned and hence are expected to contribute to a high score. The second pair of sequences shows a possible individual that has been found at some point in the search which has a decent score, but which it is hoped can be improved on. The four deletions in the database sequence are the correct amount of this segment of the alignment, but they are misplaced and hence only the first and last parts of the alignment are correct. The third pair of sequences shows what might be accomplished if simultaneous insertions and deletions are possible. The three insertions (dots in the consensus sequence) and three deletions (new dots in the database sequence) have effectively shifted a part of the database sequence in the middle (underlined with * characters) with respect to the consensus and brought 13 new positions into correct alignment even though neither the insertions nor the deletions are exactly correct. The fourth pair of sequences shows what happens if only the insertion is done. The new correctly aligned segment in the middle is picked up at the expense of losing the end correctly aligned segment. As shown in the figure, this may have close to no effect on the score, but since there is much more of the original sequences not shown to the right in the figure, the result is probably a large decrease in score. The last pair of sequences shows what happens when the deletion is made and it is clear that the score will decrease.

The conclusion that can be drawn from Figure 9 is not only that the possibility of multiple mutations should be frequently allowed after the initial generation, but that there ought to be a bias toward pairs of compensating mutations. By compensating mutations is meant an insertion and deletion pair with the same magnitude effect on the sequence length (and zero overall effect on length). The third case in Figure 9 is an example of compensating mutations with magnitude three.

Figure 9 Simultaneous vs. sequential insertion/deletion (underlined symbols are correctly aligned)

```
consensus:
   UGCCUUAAACUUAUGAGUAAGGAAAAUAACGAUUCGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGACGAAUUUUU
true alignment:
2  AGCCUCAAACUUAAGGGUAAGGAAAAUAAUGAUUGCAGCAG.UGCCUGAGUCCUG.CUGCUAAUG..AGAUGAAUUUUU

consensus:
   UGCCUUAAACUUAUGAGUAAGGAAAAUAACGAUUCGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGACGAAUUUUU
Suboptimal initial alignment:
2  AGCCUCAAACUUAAGGGUAAGG....AAAAUAAUGAUUGCAGCAGUGCCUGAGUCCUGCUGCUAAUGAGAUGAAUUUUU

consensus:
   UGCCUUAAACUUAUGAGUAAGGAAAAUAACG...AUUCGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGACGAAUUUUU
simultaneous insertion and deletion:
2  AGCCUCAAACUUAAGGGUAAGG....AAAAUAAUGAUUGCAGCAGUGCCUGAGUCCUGCUGCU...AAUGAGAUGAAUUUUU
portion of database sequence experiencing shift relative to consensus:
                                ********************************

consensus:
   UGCCUUAAACUUAUGAGUAAGGAAAAUAACG...AUUCGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGACGAAUUUUU
insertion only:
2  AGCCUCAAACUUAAGGGUAAGG....AAAAUAAUGAUUGCAGCAGUGCCUGAGUCCUGCUGCUAAUGAGAUGAAUUUUU
portion of database sequence experiencing shift relative to consensus:
                                ********************************************

consensus:
   UGCCUUAAACUUAUGAGUAAGGAAAAUAACGAUUCGGGGUGACGCCCGAAUCCUCACUGCUAAUGUGAGACGAAUUUUU
deletion only:
2  AGCCUCAAACUUAAGGGUAAGG....AAAAUAAUGAUUGCAGCAGUGCCUGAGUCCUGCUGCU...AAUGAGAUGAAUUUUU
portion of database sequence experiencing shift relative to consensus:
                                                    ******************
```

The GA just described is suggested as a prefilter for the reduction of the database by many orders of magnitude to only those positions that hold promise of generating large scores. The remaining database can then be processed using the limited classical algorithm on Section 4 to get optimal alignments of the database to the covariance model.

6 Conclusion

An analysis of the search space usage by actual ncRNA family members in the classical dynamic programming method for covariance model database search with a single maximum length parameter *D* has be shown to be potentially very inefficient. This is due to failure to use the fact that node-level best scores are strongly clustered about zero length deviation. Limiting the search space in the sequence length dimension at the covariance model node level could significantly reduce computation time with very little chance of missing true ncRNA family members.

The clustering of observed best-score sequence lengths about zero length deviation also indicates where to begin a non-exhaustive search that is not based on dynamic programming. Such non-exhaustive searches allow the possibility of early exit at a given database position if the returned scores are not promising, allowing the algorithm to move on more quickly to positions that show more initial promise. Simply gradient search from an ungapped alignment starting point is unlikely to succeed due to many local minima in the fitness landscape. Search methods that allow simultaneous insertions and deletions and those that allow portions of different solutions at different locations in the alignment are more likely to succeed. Genetic algorithm searches have both these properties.

Much remains to be done in terms of fully characterising the statistics of search space usage by all non-coding RNA families and exploring the full range of possible non-exhaustive search methods for covariance models. However, the potential of such non-exhaustive searches as prefilters to reduce the relevant database size for full dynamic-programming based covariance model search is very good.

Acknowledgement

This work was supported in part by the US National Institutes of Health under Grant P20 RR016454 of the INBRE Program of the National Centre for Research Resources.

References

Chomsky, N. (1959) ‘On certain formal properties of grammars’, *Information and Control*, Vol. 2, pp.137–167.

Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998) *Biological Sequence Analysis*, Cambridge University Press, Cambridge, UK.

Eddy, S. (1998) ‘Profile hidden Markov models’, *Bioinformatics*, Vol. 14, pp.755–763.

Eddy, S. and Durbin, R. (1994) ‘RNA sequence analysis using covariance models’, *Nucleic Acids Research*, Vol. 22, pp.2079–2088.

Eidhammer, I., Jonassen, I. and Taylor, W. (2004) *Protein Bioinformatics*, Wiley, Hoboken, NJ.

Gesteland, R., Cech, T. and Atkins, J. (2006) *The RNA World*, 3rd ed., Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, M., Eddy, S. and Bateman, A. (2005) 'Rfam: annotating non-coding RNAs in complete genomes', *Nucleic Acids Research*, Vol. 33, pp.D121–D124.

Mount, D. (2001) *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.

Yadgari, J., Amir, A. and Unger, R. (2001) 'Genetic threading', *Constraints*, Vol. 6, pp.271–292.