9-1-2006

Covariance Searches for ncRNA Gene Finding

Jennifer A. Smith
Boise State University

Covariance Searches for ncRNA Gene Finding

Scott F. Smith, *Senior Member IEEE*
Department of Electrical and Computer Engineering
Boise State University
Boise, Idaho 83725-2075 USA
sfsmith@boisestate.edu

***Abstract*-The use of covariance models for non-coding RNA gene finding is extremely powerful and also extremely computationally demanding. A major reason for the high computational burden of this algorithm is that the search proceeds through every possible start position in the database and every possible sequence length between zero and a user-defined maximum length at every one of these start positions. Furthermore, for every start position and sequence length, all possible combinations of insertions and deletions leading to the given sequence length are searched. It has been previously shown that a large portion of this search space is nowhere near any database match observed in practice and that the search space can be limited significantly with little change in expected search results. In this work a different approach is taken in which the space of starting positions, sequence lengths, and insertion/deletion patterns is searched using a genetic algorithm.**

I. Introduction

Covariance models (CMs) are used to search nucleotide databases for genes associated with new members of known non-coding RNA (ncRNA) families. The model parameters are estimated from a group of nucleotide sequences which have been determined to be related. The estimated model is then used to search for similar patterns in nucleotide databases that ideally would include sequences of entire chromosomes. The computational burden of parameter estimation is minor, but that of database search is extremely large.

Covariance models can be thought of as an extension of profile hidden Markov models (HMMs) [1]. The major difference is that the CM can describe interactions between positions in the sequence whereas the HMM can only describe conservation at each sequence position individually. The CM is said to have a context-free grammar in the Chomsky [2] hierarchy of transformational grammars and the HMM is at the next lower level with a regular grammar. The fact that an HMM can be viewed as a restricted CM has been used to design pre-filters that can remove portions of the database from consideration by the CM [3]. Given the parameters and score threshold for the CM search a set of parameters and score threshold can be computed for an HMM that guarantees that the HMM search will return sequence regions which are a superset of those that would be returned by the CM. Since it is much faster to score a sequence against an HMM than a CM, the overall search is sped up by using an initial pass with the HMM pre-filter followed by the full CM search. However, even with pre-filtering the use of covariance models is extremely slow. Without pre-filtering the use of a CM is usually infeasible.

The need to model interactions between ncRNA sequence positions, but not protein or protein-coding gene sequences, comes from differences in the way the two evolve. Proteins tend to have a fair amount of primary sequence conservation at least in some portions of the sequence. When combined with amino-acid substitution matrices [4-5] this primary conservation becomes even more evident. Non-coding RNA sequences also exhibit some level of primary sequence conservation, but not nearly to the degree as proteins or protein-coding genes. The function of the ncRNA molecule is mostly determined by the intra-molecular base pairing arrangement of its nucleotides. There is little evolutionary pressure to conserve a particular base at a particular sequence position as long as a compensating variation is made elsewhere to maintain base pairing (hence the word covariance in covariance model).

Scoring of covariance models is normally done using a dynamic programming method. Regular-grammar-based models such as the HMM and Smith-Waterman [6] use dynamic programming by adding a symbol to one end of an existing solution. In order to allow intra-molecular interaction, the CM uses dynamic programming to add a symbol either to one end or to both ends of an existing solution simultaneously. The model also allows a bifurcation (joining) operation for two contiguous existing solutions. This process is used to find the scores for every starting position in the database sequence and every possible length d from that starting position. In order to make the calculation tractable, the search over lengths from the starting position is limited to a user defined maximum D. This maximum is normally chosen to be about one and one-half to two times as large as the length of the consensus sequence of the sequences used to build the CM. Fitting a database sequence of length d to the CM can be done using any number of insertions and deletions at any positions in the sequence such that the net length matches that of the model. The standard dynamic-programming method searches all possibilities.

It has been noted that much of the search space for the standard CM search is nowhere near any observed solution [7]. In particular, the values of d searched at different phases of the dynamic programming calculation are often very far from those ever observed in practice. A reduction in computational burden of a couple of orders of magnitude is

1-4244-0623-4/06/$20.00 ©2006 IEEE

usually available by simply constraining this dimension of the search. This paper takes a different approach, where the search instead proceeds from scoring the ungapped (no insertions or deletions) database sequence at each starting position with length *d* exactly equal to the length implied by the model. The search is then expanded about promising starting locations through the addition and/or removal of insertions and/or deletions in the database sequence. The hope is to find good solutions much more quickly than the standard dynamic programming solution method. An added benefit is that there is no pre-defined cutoff *D* on allowed sequence lengths in the database. Thus this method may potentially find remote ncRNA family members with large net numbers of insertions relative to the CM. The search is done with a genetic algorithm (GA) since the shape of the fitness landscape is largely unknown and therefore a search algorithm that can avoid getting caught in possible local minima is desired.

The paper is organized as follows. In Section II, a very short review of how covariance models are estimated and used for database search is undertaken. The GA-based covariance search (CS) algorithm is described in Section III and takes the parameters of the CM as a starting point. A test of the CS on a well-known ncRNA family is examined in Section IV. Section V gives some concluding remarks.

II. Covariance Models

Covariance models are described much more thoroughly in Chapters 9 and 10 of [8]. The overview here is given so that those unfamiliar with the structure of a CM can better understand the differences between the standard scoring method and the method of Section III.

A. Estimation of Covariance Model Parameters

The estimation of covariance model parameters can be thought of as if it was a two-step process where first one generates a multiple alignment of the known ncRNA family sequences and then uses the multiple alignment to choose the states of the model and calculate the model parameters. In fact, the multiple alignment and parameter estimation are usually done jointly, but for expositional purposes a two-step process will be assumed here. Since the procedures for generating multiple alignments are well known (see for example [9]), they will not be presented here.

Given a multiple alignment, one can use a number of metrics to decide whether a column of the alignment is a conserved position or not. This metric could be as simple as choosing columns with less than fifty percent gap characters or something more complex involving the distribution over all possible symbols. The conserved columns end up being represented by symbol-emitting nodes in the final CM. In addition to the multiple alignment, one needs an annotation of the base pairing pattern of the columns (a secondary structure annotation). This can be derived either from experiment or from secondary structure prediction using, for example, the dynamic-programming-based Zucker algorithm [10] or the genetic-algorithms-based P-RnaPredict [11]. This pairing information will determine whether the conserved column is represented in the model as a single-symbol emitting node or a paired-symbol emitting node.

Single-symbol emitting nodes come in two flavors, those that emit a symbol on the left (5' end) of an existing solution and those that emit a symbol on the right (3' end). These nodes are referred to as L and R nodes respectively. The paired-symbol emitting nodes are called P nodes and emit one character on each end of an existing solution. There are also three types of non-emitting nodes in covariance models: the start (S) node, the end (E) node, and the bifurcation (B) node. The CM can be represented as a binary tree with the root at the top. The B node is found at all branch points in the tree. The E node is found at the bottom of all terminal branches. The S node is found at the root and every B node has two S nodes for children. The non-emitting nodes allow the emitting nodes to be structured into a tree and allow a place to add additional sequence insertion points that do not fit conveniently in the emitting nodes.

Figure 1 shows a portion of the multiple alignment for the U12 snRNA. Only a small portion of the multiple alignment columns are shown. This ncRNA family will be discussed more fully in Section IV. The consensus sequence and consensus secondary structure are shown near the bottom of the figure. Those columns without a gap symbol (.) as the consensus symbol will be assigned to an emitting node (L, R, or P) in the CM. Those with an unpaired symbol (-) in the consensus secondary structure will be assigned to either an L or an R node and those with a paired symbol (< or >) will be assigned as a pair to a P node. The < symbol means that the other sequence position of the pair is to the right and the symbol > that the other position of the pair is to the left. It is often (but not always) the case that an unpaired node can be assigned to either an L or and R node. By convention, the L node is always chosen if either type node is possible.

The use of < and > for secondary structure notation is unambiguous only because pseudoknots have been ruled out. If the structure to be modeled actually contains a pseudoknot, then the model builder must choose to ignore some of the base pairing information and label some of the columns as unpaired that are actually paired. This results in loss of information in the model and therefore some loss of model power. Luckily, most ncRNA families do not contain pseudoknots. The inability to handle pseudoknots comes from using a context-free grammar rather than the next higher level in the Chomsky hierarchy (a context-sensitive grammar). The additional complexity of a general context-sensitive grammar would make computation of the model scores infeasible.

The node structure tree of the multiple alignment fragment shown in Figure 1 is given in Figure 2. The node labels shown in the boxes correspond to the "CM nodes" listing at the bottom of Figure 1. Three dots in the tree are an abbreviation that means there are a sequence of nodes of the same type with consecutive node numbers that are not shown. A triangle on the side of a box indicates that a symbol is emitted on the given side of a node. The triangles are labeled with the consensus nucleotide for the emission. The L and R nodes are

in fact capable of emitting any nucleotide and the P nodes are capable of emitting any pair of nucleotides. The labeling simply shows that nucleotide emitted with highest probability. The alignment fragment shown in Figure 1 is the portion on the 5' (left) end of the alignment and the actual U12 alignment continues far to the right. Only the left half of the second stem (P30-P33) is shown which is why these nodes have been labeled with a question mark in Figure 2. Any additional conserved columns enclosed between the left half-stem shown and the right half-stem (not shown) would follow below node P33 in the tree. Additional stems (sequences of P nodes) to the right of the alignment column associated with the right half of P30 would require additional bifurcations somewhere above B3. The full U12 tree requires four bifurcations.

```
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACAACU
AGCCUCAAA....CUUAAGGG.UAAGGAAAAUAAUGAUU  more
UGCCUUAAA....AUUAUAAG.UAAGGAAAAUAAUGAUU  to
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU  right
UGACUUA.A....CUU...AGCUAAGGAAAAUAAUGGUU  not
UGCCUUAAA....AUUAUAAG.UAAGGAAAAUAAUGAUU  shown
GGUAUUAAGAGUUCUUAUGAG.UAAGGAAAAUAACGAUU
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU
UGUCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU
UGUCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU

consensus:
--<<<<<<-....--------.>>>>>--------<<<<
UGCCUUAAA....CUUAUGAG.UAAGGAAAAUAACGAUU

CM nodes:
LLPPPPPLL....LLLLLLLL.PPPPPLLLLLLLLPPPP
       11    11111111      222222223333
125678901    23456789 98765234567890123
```

Fig. 1. Multiple alignment fragment for CM example.

The node structure of the CM describes the model consensus, but it is also necessary to allow database sequences with insertions and deletions relative to the consensus. This is accomplished by the internal organization of the CM nodes. CM nodes are composed of one or more internal states. Each node has a state which mirrors the consensus function of the state, so L nodes contain an L state and P nodes contain a P state, etc. Insertion is allowed by having extra L and/or R states within some node types. For instance, all L nodes have one L state for the consensus left emission (called a match left or ML state) as well as an extra L state than can be visited multiple times allowing one or more insertions (called an insert left or IL state). The transition probabilities into IL states are assigned such that there is a score penalty for visiting the IL state when compared to choosing a path that does not visit the insertion state. There are also delete states (D states) added to the emitting nodes to allow the match state to be bypassed.

The parameters of a covariance model include all of the state transition probabilities between states and all of the emission probabilities for symbol-emitting states. Conceptually, these could be found from the multiple alignment by counting the frequency of occurrence of each event in the alignment. For instance, the four emission probabilities assigned to A, C, G, and U respectively in the ML state of node L1 could be found by observing that there is one A, no C, one G, and 12 Us in the first alignment column. In order to avoid an infinite score penalty if the database sequence were to actually contain a C in that position, a pseudo-count is normally added. The probability calculation then proceeds as if the counts had been 2, 1, 2, and 13 for A, C, G, and U respectively. This results in probabilities of 2/18, 1/18, 2/18, and 13/18 assigned to the four emission probabilities. The actual values used in the models are logarithms of probabilities so that the scoring program can add log probabilities rather than multiply linear probabilities. The emission probabilities are also often corrected for nucleotide composition bias. Transition probabilities could be generated by observing the frequency with which the alignment sequences visit the various states by observing the insertions and deletion in the multiple alignment with respect to the consensus.

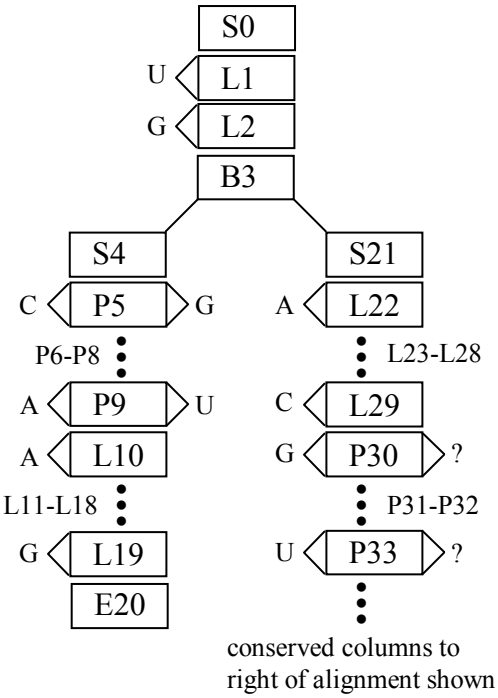

Fig. 2. Node structure of example CM fragment.

B. CM Database Scoring Using Dynamic Programming

Given a database sequence that may be very long compared with the potentially embedded ncRNA gene, one would like to know the probability that each nucleotide position is part of the ncRNA family of interest. Such a set of scores can be determined by dynamic programming by starting at the end nodes at the bottom of the CM tree and working up toward the root start node (S0). The maximum score over all examined sequence lengths of the root start state inside the root start node is the overall score for the given start position in the database sequence with respect to the CM. The end nodes contain only a single state called an end or E state. The E states simply set the score to 0 for all sequences of length 0

and minus infinity for all sequences longer than 0. Moving up the tree additional nucleotides are added on the right and/or left ends of the existing solution represented by the subtree below the node being evaluated. The result is a calculation of a score at every state for every possible start position and every possible number of nucleotides extending from the start position. In order to make this tractable, a maximum extension length *D* is chosen beyond which scores are not calculated.

III. Covariance Search

It has been previously observed that a large portion of the search space covered buy the dynamic programming method (all start locations and all lengths up to *D* at each of these locations) is nowhere near any sequence observed in practice [7]. One way to speed up CM search is to limit the search sequence length at each model state to a region around the length of the consensus sequence segment represented by the subtree below the given state. This reduces the search space in the sequence length dimension, but still requires the same amount of computation at every start position. This work takes a different approach based on a genetic algorithm (GA) that focuses the computational effort on regions of the search space with both promising start positions and sequence lengths.

A. Representation

In order to use a GA with crossover, it is convenient to have a fixed length representation of how a subsequence of the database is to be fit to the CM. Such a representation already appears in the protein threading literature as presented by Yadgari et al [12]. This representation uses a vector of non-negative numbers with length equal to the model length. A vector of all ones implies that the database subsequence is to be aligned with the model with no insertions or deletions. An entry of 0 anywhere in the vector means that the database has no symbols assigned to the given model location (the database has a deletion at this position with respect to the model). An entry of two or more means that one or more symbols from the database are inserted to the right of the model position.

For use in covariance models, one element of the representation vector will be assigned to each conserved column of the multiple alignment (in order from left to right). Single-emission nodes (L and R) are associated with a single vector element and P nodes are associated with two vector elements. An additional element is added to the vector which is a non-negative integer representing the start position within the database sequence.

B. *Scoring*

Given a representation vector and the parameters of a CM, a database sequence may be scored as follows. The location of the database symbol to be matched to a consensus column of the model is given by $S + sum(1:C) - 1$, where S is the start location and C is the conserved column number. The *sum(a:b)* operation adds up to vector elements starting at element *a* and ending at element *b*. For unpaired columns, the database symbol is used as an index to select one of four log match probabilities to add to the score. For paired columns, the database symbol is found for both columns of the pair and the two symbols index one of 16 match probabilities for addition with the score. The score is initially set at zero and the additions take place for each unpaired column and each pair of columns.

The transition probability for the path through the delete state compared to the path through the match state represents a node-dependent deletion penalty. The representation vector is scanned for zeros and the penalties associated with the location of the zeros are applied to the score. The transition probability of the path through both match state and insert state relative to that of the match state alone represents an insertion penalty for a single insertion. The transition probability of the path which visits the insert state twice and the match state once relative to the match state once is the two insertion penalty. Visiting the insert state more than once requires making a self-loop transition which has a fixed transition probability, hence the insertion penalties are affine with the penalty for the first insertion possibly different than the second, but all subsequent additional penalties the same. The representation vector is scanned for values greater than one and the associated penalties applied.

C. Genetic Algorithm

An initial population for the GA is found by searching all start positions in the database sequence and calculating the score for the ungapped alignment of the database to the model at each position. The start positions with the highest scores are selected for the initial population. The representations for the initial population are therefore each a vector of all ones concatenated with a high-scoring start position. It is often the case that several high-scoring upgapped alignments are found in close proximity since insertions or deletions near the center of the model will result in good matches to one model part and then the other model part as the database sequence is brought into alignment with the two pieces. Since the mutation operation described below will allow for small movements in the start position, two initial population individuals in close proximity in start position are likely to converge to the same solution. In order to get more initial diversity, the initial population selection works from the highest score down and skips over high-scoring starting positions that are within a user-specified distance of a higher-scoring starting position. In the results in the next section, a constant population size of 100 and a start position distance minimum of 3 worked well.

Subsequent generations are selected by keeping top scorers unchanged (elitism) and using both crossover and mutation. Since the database sequence might be as long as an entire chromosome, there may be many members of the ncRNA family in a single database sequence. In order to keep multiple high-scoring local maxima in the search space alive, a fairly large number of elite are retained unchanged. These elite are also selected such that they are not too close together in starting position. The same method that was used in selecting the initial population is used to select the elite to retain. A

minimum distance of 3 was used to retain 30 elite in the test results in the next section.

Crossover is accomplished by random uniform selection of a crossover point in the representation. The fixed length representation assures that the insertion/deletion pattern swap is associated with the same multiple alignment columns in both parents. Crossover is expected to be a very useful operation in this application since even a casual look at almost any ncRNA multiple alignment makes clear that similar insertion/deletion patterns occur in several family members in the same region. This is often true in multiple regions even though the set of family members in the relation may differ. This is an example of the GA being more than simply inspired by biology, but rather a reasonable model of the true underlying process generating the data to be searched.

Mutation is a bit more complicated than is usual in a GA. It is desirable to allow the positional window onto the database sequence being examined to shift its start position as well as its length and internal insertion/deletion pattern. This can be done by simply treating the start position element of the representation vector like any other element. However, this element would be mutated very infrequently relative to the internal shifts if the consensus sequence was very long. Rather than just increasing the mutation probability of the start position element, the start position is modified along with the internal insertion/deletion one-half of the time.

```
no mutation
data:  UGCCUUAAAUUAUGAGUAAGGAAAAUAACAACU
model: UGCCUUAAACUUAUGAGUAAGGAAAAUAACGAU
mutation pt:    *

insert - no start change
data:  UGCCUUAAAUUAUGAGUAAGGAAAAUAACAACU
model: UGCCUUAAAC.UUAUGAGUAAGGAAAAUAACGAU

delete - no start change
data:  UGCCUUAAA.UUAUGAGUAAGGAAAAUAACAACU
model: UGCCUUAAACUUAUGAGUAAGGAAAAUAACGAU

insert - compensating start change
data:   UGCCUUAAAUUAUGAGUAAGGAAAAUAACAACU
model: UGCCUUAAAC.UUAUGAGUAAGGAAAAUAACGAU

delete - compensating start change
data:  UGCCUUAAA.UUAUGAGUAAGGAAAAUAACAACU
model:  UGCCUUAAACUUAUGAGUAAGGAAAAUAACGAU
```

Fig. 3. The four mutation cases (unchanged portion of alignment underlined).

Figure 3 shows the idea behind this mutation strategy. There are four cases: insertion, deletion, insertion with compensating start shift, and deletion with compensating start shift. The first and second cases keep the database subsequence to the left of the mutation point aligned with the model the same way as before the mutation while shifting the database subsequence to the right of the mutation point with respect to the model. The two compensated start point cases shift the left subsequence while keeping the right subsequence fixed with respect to the model. These mutation cases are applied with equal probability in the results section. Mutations take the form of incrementing or decrementing a value by 1, where the mutation is abandoned if the vector element would become negative (since only non-negative values are meaningful).

IV. U12 snRNA Search

As a test case the GA-based covariance search proposed in the preceding section was run against a database containing the fourteen known members of the U12 small nuclear RNA (snRNA) family available at the Rfam website [13-14]. The U12 functional RNA molecule is part of the minor spliceosome used to remove introns from pre-mRNA. It has a function similar to that of the U2 snRNA in the major spliceosome, but the two families have quite different sequences. The Rfam (RNA family) accession number RF00007 indicates that it is one of the oldest known non-coding RNAs. The consensus length of the U12 family is 149. The CM was estimated from 7 "seed" sequences taken from the literature (1 in chicken, 1 in mouse, and 5 in human) and used to find an additional 7 family members (all in mouse).

A database sequence of length 15880 bases was constructed which contains all fourteen U12 sequences with several randomly selected ncRNA sequences from other families interspersed between each U12 sequence and the next. The covariance search was run for 1000 generations with a population size of 100 and other parameters as described in Section III.C. Table I shows the results. The "true start" column of the table shows where the first nucleotide of the family member sequence taken from Rfam is located in the constructed sequence. "Start found" indicates the start location of highest-scoring individual in the final generation that was in close proximity (within 20) of the true start. The score rank gives the placement of the individual in a ranked list of scores of the final generation and the associated score follows in the next column. Finally the EMBL accession number of the sequence and sequence position numbers where Rfam originally found the ncRNA family member is given.

Of the fourteen true family members embedded in the test sequence, twelve were found. Ten sequences were located exactly and two others where found to within 2 and 8 bases respectively. The top five ranked representations were distinct family members. All of the top 28 individuals are either listed in the table or had a start position with one place of the ones listed in the table. The others are not really false positives, they are just minor variants found in the last round from mutation and crossover.

The scores of the 100 individuals in the final generation are shown in Figure 4. There is a significant drop at about the 38th individual from around 80 to about 50. The scores have units of bits, so these scores are far above anything that is likely to be produced by a random matching of sequences. It is likely that the cutoff score for declaring a database sequence region to be a hit would be no more than 80, so all 12 sequences found in Table I would likely also be found in a search of a much larger database.

TABLE I
ALRORITHM PERFORMANCE ON U12 GENES

True Start	Start Found	Score Rank	Score	Accession Number / Nucleotide Positions
446	**none**	-	-	L43844.1/2-149
1039	1039	10	150.1	AC087420.4/142608-142466
2475	**2477**	21	136.0	AC112938.11/234142-234291
3858	3858	11	150.1	AL591952.9/131760-131611
6096	6096	18	139.0	AL669944.8/2483-2625
7406	7406	5	162.8	AC133939.4/22042-22191
8196	**none**	-	-	AC132590.3/81080-80927
8880	8880	13	148.4	AL772347.6/146375-146226
9705	9705	1	165.9	L43843.1/2-150
10774	10774	4	165.1	L43846.1/332-480
11624	11624	9	155.6	J04119.1/2-150
12428	**12436**	28	118.4	L43845.1/358-512
13493	13493	3	165.5	Z93241.11/76642-76790
14615	14615	2	165.9	AL513366.11/57717-57871

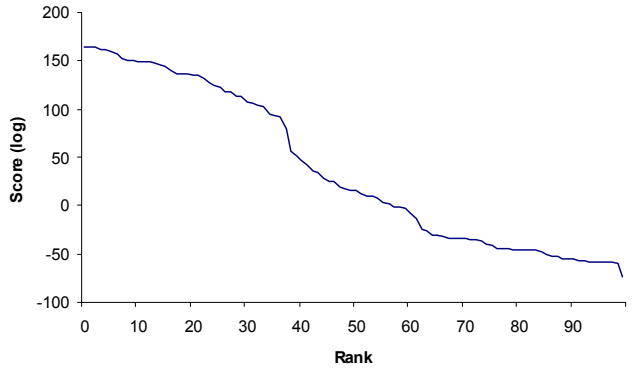

Fig. 4. Scores of individuals in final generation.

Figure 5 shows the database starting location value in each individual's representation in the final generation, where the individuals have been resorted in ascending order of starting location value. It is clear from the plot that the GA has converged on solutions close to the twelve starting values found in Table I. There are no individuals searching anywhere near the two missed locations (446 and 8196). There is a general (but not perfect) trend towards having more individuals searching around those positions where the highest scores have already been found, which is to be expected. However, the method used to maintain position diversity during elite selection appears to work since the population does not appear to converge around a single location.

The mean values over the 100 individuals of the final generation at each conserved multiple alignment position are shown in Figure 6. Of particular note is the spike at alignment column 103, where the mean representation vector value is 1.6 in a region surrounded with much less activity. The multiple alignment of the fourteen sequences from Rfam shows that five of the sequences have an insertion relative to the consensus of length one after column 103. The algorithm has found this division point exactly even though it only has access to the unaligned sequences. Two of the sequences (starting at 12428 and 14615 in the database sequence) have an insert of length 6 after position 33 relative to the consensus (while the other 12 have no insertion). There is some evidence that the GA is discovering this with the mean value of 1.09 for positions 26, 27, and 28 (just a bit too far left). The score for the U12 family member at 14615 seems to be using this since the start position is dead on, but the best start position of 12436 rather than the true start of 12428 indicates that the start is being shifted to absorb the extra symbols rather than the insert for this family member. There is a fair amount of activity in the multiple alignment to the left of position 33 and this shows in the mean representation value in Figure 6. Mean representation values in this region do not match too well with the activity in the multiple alignment. This may be due to the fact that getting the correct number of insertions and deletions near the middle of the sequence is more critical for getting a high score than it is near the ends. For example, if the database sequence has a single insertion at the half way point, then an ungapped alignment can only match half the sequence. If a single insertion appears at the next to last nucleotide, only a single position can not be aligned with an ungapped alignment.

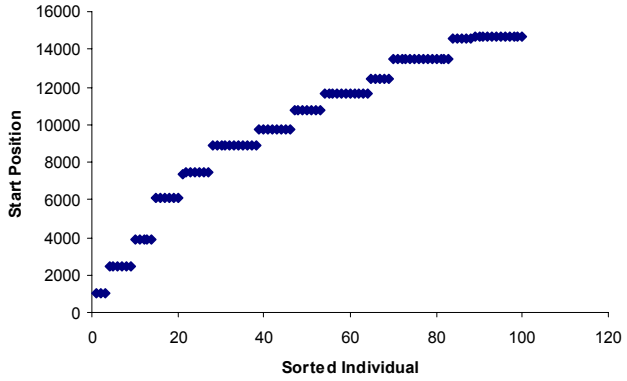

Fig. 5. Sorted start positions of individuals in final generation.

Figure 7 shows the mean score at each generation of the best thirty individuals. The GA looks like it has mostly converged after about the 600th generation. Since there are thirty individuals retained at each generation as the elite, one might wonder why this curve is not monotonically increasing. If the algorithm was not designed to also maintain start position diversity, this curve would in fact be monotonic. The negative variations are an artifact of new, slightly better, solutions coming into the elite that displace two existing members of the elite by having a start position between the two.

The code for the covariance search was written in Matlab for ease of prototyping. If the code was converted into C it would be expected to run at least as fast and probably much faster. Even though the implementation is likely not very efficient, it took about 100 seconds to run 1000 generations with 100 individuals on the 15880 base dataset. For comparison, the U4, U5, and U6 covariance models of 137,

118, and 106 consensus length respectively were found to take 1258, 1081, and 563 days of CPU time to run against an 8-gigabase database on a Pentium machine similar to the one used in this study [3]. The covariance search code has already run 2.7 times as fast as the standard code on the U4 model. Since the U12 model is of length 149, it is likely to take even longer than U4 to run with the standard code. The standard code (Infernal) [15] is a well tuned compiled C program and is not likely to get much more efficient. The code for the proposed algorithm is no where near optimized. It is also likely that using a stopping criterion would significantly reduce the required number of generations.

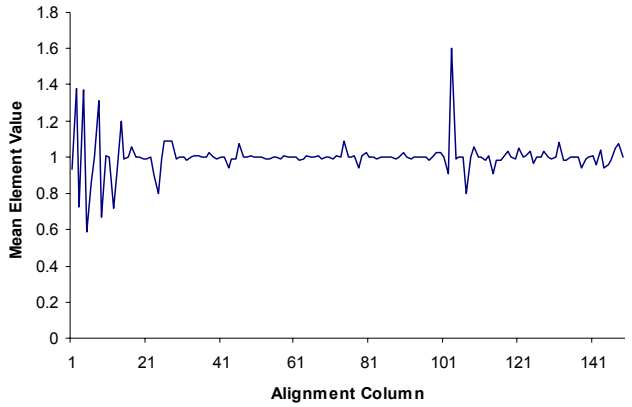

Fig. 6. Mean representation element values in final generation.

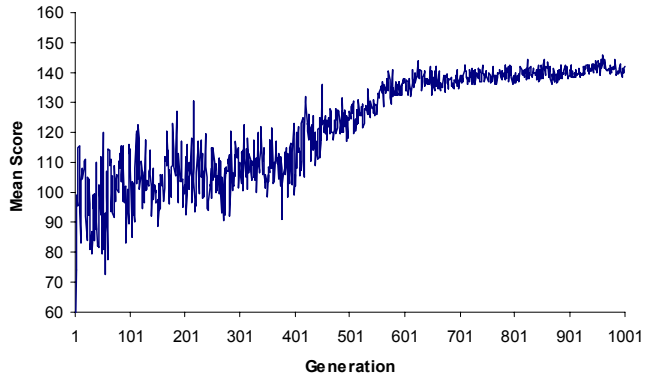

Fig. 7. Mean score of best 30 solutions by generation.

The Matlab source code and the example database are available on the web at [16].

V. Conclusions

The proposed GA-based covariance search algorithm takes advantage of the fact that the standard dynamic programming algorithm for database search using covariance models calculates a large fraction of its scores in regions of the search space that are highly unlikely. Previous work has shown how to reduce the inefficiency of this search in the sequence length dimension, but has done nothing to concentrate computational effort about any particular point in the database position dimension. This paper corrects that deficiency by using a genetic algorithm. The results on a single ncRNA family (U12) look promising. However, much work remains to be done. More functional RNA families need to be explored to determine whether these results really are general. Also, no attempt has yet been made to fine tune the parameters, such as population size, stopping criteria, numbers of elite retained, or level of mutation and crossover when creating new individuals. Finally, the GA-based method has not yet been compared to other possible search algorithms such as simulated annealing or simple hill climbing.

Acknowledgment

The project described was supported by NIH Grant Number P20 RR016454 from the INBRE Program of the National Center for Research Resources.

References

[1] S. Eddy, "Hidden Markov Models," *Current Opinion in Structural Biology*, 6, pp. 361-365, 1996.
[2] N. Chomsky, "On Certain Formal Properties of Grammars," *Information and Control*, 2, pp. 137-167, 1959.
[3] Z. Weinberg and W. Ruzzo, "Faster Genome Annotation of Non-coding RNA Families Without Loss of Accuracy," *Int. Conf. on Research in Computational Molecular Biology*, pp. 243-251, 2004.
[4] S. Henikoff and J. Henikoff, "Amino Acid Substitution Matrices from Protein Blocks," *Proceedings of the National Academy of Sciences*, 89, pp. 10915-10919.
[5] M. Dayhoff, "Survey of New Data and Computer Methods of Analysis," in *Altas of Protein Sequence and Structure*, 5, National Biomedical Research Foundation, 1978.
[6] T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, 147, pp. 195-197, 1981.
[7] S. Smith, "Acceleration of Covariance Models for Non-coding RNA Search," *International Conference on Bioinformatics and Computational Biology*, in press, 2006.
[8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*, Cambridge University Press, 1998.
[9] D. Mount, *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, 2001.
[10] M. Zucker, "Computer Prediction of RNA Structure," *Methods in Enzymology*, 180, pp. 262-288, 1989.
[11] K. Wiese, A. Hendriks, A. Deschênes, and B. Youssef, "Significance of Randomness in P-RnaPredict - A Parallel Algorithm for RNA Folding," *IEEE Congress on Evolutionary Computation*, 2005.
[12] J. Yadgari, A. Amir, and R. Unger, "Genetic Threading," Constraints 6, pp. 271-292, 2001.
[13] Rfam website, http://rfam.wustl.edu.
[14] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. Eddy "Rfam: An RNA Family Database," *Nucleic Acids Research*, Vol. 31, No. 1, pp. 439-441, 2003.
[15] Infernal Users Guide, http://www.genetics.wustl.edu/eddy/infernal/.
[16] Source Code and Database for GA-based Covariance Search, http://coen.boisestate.edu/ssmith/BioHW/CompCode/GACM/GACM.htm.