4-1-2007

# RNA Gene Finding with Biased Mutation Operators

Jennifer A. Smith
*Boise State University*

**Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2007)**

# RNA Gene Finding with Biased Mutation Operators

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 genetic algorithms for non-coding RNA gene finding has previously been investigated and found to be a potentially viable method for accelerating covariance-model-based database search relative to full dynamic-programming methods. The mutation operators in previous work chose new alignment insertion and deletion locations uniformly over the length of the model consensus sequence. Since the covariance models are estimated from multiple known members of a non-coding RNA family, information is available as to the likelihood of insertions or deletions at the individual model positions. This information is implicit in the state-transition parameters of the estimated covariance models. In the current work, the use of mutation operators which are biased toward selection of insertions and deletions at model positions with low insertion or deletion penalties is examined in hopes of speeding up convergence. The performance of the biased and unbiased mutation operators is compared. Both biased and unbiased genetic algorithms are also compared to a steepest-descent algorithm, which is a comparison lacking in prior work.**

## I. Introduction

Covariance models (CMs) are an extension of profile hidden Markov models (HMMs) which allow for RNA intermolecular base pairing information to be captured [1,2]. The HMM can model primary sequence homology, which makes it a good model for families of protein and protein-coding gene sequences. However, most of the information contained in an alignment of non-coding RNA sequences is in the consensus base-pairing secondary structure. It is necessary to use a model which captures the joint probability distributions of the two base-paired positions (such as the CM) rather than model that only uses the marginal distributions of the individual positions (such as the HMM) to make use of this secondary structure conservation information.

While the CM is a good model for RNA gene search, it is much more computationally demanding than the HMM when dynamic programming is used to find optimal scores. This makes the use of a dynamic-programming CM database search very expensive. The alternative of using a genetic algorithm (GA) to search the space of database start positions, database sequence lengths, and insertion/deletion patterns has been proposed [3]. While this approach seems promising, the speed of convergence is probably slower than it could be due to the fact that insertion/deletion pattern space is explored without regard to the probabilities of these insertions or deletions. In this work, the exploration of new insertion and deletion locations with uniform probability will be referred to as an unbiased mutation operator. The alternative of choosing locations with lower insertion or deletion penalties with higher probability will be called a biased mutation operator.

Candidate solutions for putative RNA genes in the database take the form of a database start position and an insertion/deletion pattern of the database symbols with respect to the consensus sequence of the RNA family model. The database is first scanned using the ungapped consensus model for promising database start positions. The search is then expanded about these start positions by altering the insertion and deletion pattern from the base ungapped pattern.

A comparison of the unbiased GA, biased GA, and a steepest descent algorithm is made in the final section using the same number of candidate solution evaluations in each of the three algorithms. Before evaluating the three algorithms, a discussion of how an individual candidate solution is scored (regardless of the search algorithm) is given in Section II. This is followed by a description of the three search algorithms in Section III. The algorithm comparison results are presented in Section IV and Section V contains some concluding remarks.

## II. Scoring Search Candidates

Candidate alignments in each of the search algorithms are represented by a starting position in the database and an alignment vector $V$. This vector has the same length as the consensus sequence of the RNA family model. Elements of the alignment vector are non-negative integers. An element value is 0, if the model position is to be deleted. If a vector element is 1, a database symbol is matched to the model position and no database symbols are inserted to the right of the model position. If an element value $n$ is greater than 1, a database symbol is matched to the model position and $n$-1 database symbols are inserted to the right of the model position. This representation does not allow a direct transition from model position deletions to database symbol insertions. This is not a major problem since this situation is extremely rare in real biological sequences and is often disallowed in sequence models anyway. This alignment representation has been previously used by [4] in the context of protein threading.

*A. Scoring Algorithm*

Given a candidate alignment vector *V*, it is necessary to score the quality of the vector, no matter which search method is used. The algorithm that follows implements this scoring based on a set of position-specific scoring parameters. The conversion of parameter values from the standard covariance model (CM) file entries [5] to the required position-specific values is the topic of the following subsection.

The algorithm for calculating the score contribution *S(i)* of an alignment *V* starting at a given database location for a particular model position *i*, is given in Figure 1. The index of the first model position is 1 and a position of 0 is reserved to mean no position. A vector of pair positions *P* is created such that *P(i)* = 0 if the model position *i* is a single-emission position, otherwise *P(i)* is the position paired with position *i* in the model. A matrix of database symbols *B* that are aligned to each model position is created using the database start location and the alignment vector *V*. The vector *B(i,●)* has *V(i)* valid entries, where *B(i,*1*)* is the symbol to match (if any) and *B(i,j)* are the symbols to insert to the right of the consensus position *i* for $2 \leq j \leq V(i)$. Normally, the entries in *B* would be one of four numbers associated with the four possible bases A, C, G, and T (or U).

```
If P(i) = 0             ; Unpaired position
  If V(i) = 0           ; Deleted position
    If V(i-1) = 0       ; Delete continuation
      S(i) = DC(i)
    Else                ; Delete open
      S(i) = D(i)
  Else                  ; Match single
    S(i) = M[i,B(i,1)]
ElseIf P(i) > i         ; Left position of pair
  If V(i) = 0           ; Left not present
    If V[P(i)] = 0      ; Right not present
      S(i) = D(i)       ; Deleted pair
    Else                ; Right present
      S(i) = MR{i,B[P(i),1]} ; Match right of pair
  Else                  ; Left present
    If V[P(i)] = 0      ; Right not present
      S(i) = ML[i,B(i,1)] ; Match left of pair
    Else                ; Both present
      S(i) = M{i,B(i,1),B[P(i),1]} ; Match pair
If V(i) > 1             ; At least one insert
  S(i) = S(i) + I[i,B(i,2)]   ; Insert open penalty
If V(i) > 2             ; More than one insert
  For j = 3 to V(i)
    S(i)=S(i)+IC[i,B(i,j)] ; Continuation penalties
```

Fig. 1. Scoring algorithm for one position in model.

The position-specific score parameters are contained in *D* (delete open), *DC* (delete continuation), *M* (match), *MR* (match right), *ML* (match left), *I* (insert open), and *IC* (insert continuation). Match and insertion score parameters depend on the database symbols matched or inserted. Delete score parameters do not involve a database symbol and are therefore a single value at each model position. The *MR* and *ML* parameter matrices have four values at each model position (one for each possible matched database symbol) and are only used for paired model positions, where one of the pair symbols is missing in the alignment. When a model position is paired, all sixteen values in the *M* parameter matrix are used at that position (one for each possible pair of matched database symbols). When a model position is unpaired, only the first four values in the *M* parameter matrix are used at that position. The *I* and *IC* parameter matrices have four values at each model position.

The overall score *S* of a candidate alignment vector *V* is the sum of the model position scores *S(i)* over all model positions *i*.

TABLE I
SCORING PARAMETER CALCULATIONS

Position *i* associated with a CM L node:

$M(i,DS) = t_{CM} + e_M(DS)$
$D(i) = t_{CD}$
$DC(i) = t_{DD}$
$I(i,DS) = t_{CI} + t_{IM} + e_I(DS)$
$IC(i,DS) = t_{II} + e_I(DS)$

Position *i* associated with a CM R node:

$M(i,DS) = t_{CM} + e_M[DS]$
$D(i) = t_{CD}$
$DC(i) = t_{DD}$
$I(i\text{-}1,DS) = t_{CI} + t_{IM} + e_I(DS)$
$IC(i\text{-}1,DS) = t_{II} + e_I(DS)$

Position *i* associated with a CM P node:

$M(i,LS,RS) = t_{CM} + e_M(LS,RS)$
$ML(i,LS) = t_{CML} + e_{ML}(LS)$
$MR(i,RS) = t_{CMR} + e_{MR}(RS)$
$D(i) = t_{CD}$
$DC(i) = t_{DD}$
$I(i,LS) = t_{CIL} + t_{ILM} + e_{IL}(LS)$
$IC(i,LS) = t_{IIL} + e_{IL}(LS)$
$I(i\text{-}1,RS) = t_{CIR} + t_{IRM} + e_{IR}(RS)$
$IC(i\text{-}1,RS) = t_{IIR} + e_{IR}(RS)$

*B. CM to Scoring Algorithm Parameter Conversion*

In order to convert the transition and emission scores in a standard CM model file into the form used by the algorithm in Figure 1, the equations in Table I may be used. For left emission (L) and right emission (R) CM nodes, *DS* means the database symbol being matched or inserted. For pair emission (P) CM nodes, *LS* and *RS* mean database symbols being matched or inserted on the left or right sides respectively. Transition scores are denoted *t* and emission scores *e*. The subscripts refer to states within the node or child node. Since CM bifurcation nodes are non-emitting and only bifurcation nodes have more than one child, there is only one child node in all cases in Table I. The *C* subscript denotes the consensus state within the child node. The child consensus state is the MP (match pair), ML (match left), MR (match right), B (bifurcation), and E (end) state in the P (pair emission), L (left emission), R (right emission), B (bifurcation), and E (end) node respectively. All emission score subscripts refer to that state within the node being evaluated. Transition score subscripts with a *C* imply a transition from the consensus state within the child node to a state within the node being evaluated. The transition score $t_{DD}$ is between the delete state of the child node and the delete state of the evaluated node.

All other transition scores are between states in the evaluated node.

The main thing that might at first seem odd in Table I is that some of the insertion score parameter equations have *i*-1 rather than *i* as the consensus position. The reason is that extra inserted symbols occurs first in a node before matching. For left emissions (in L or P nodes), this results in the inserted symbols being to the right of the match symbol. For right emissions (in R and P nodes), the inserted symbols are to the left of the match symbol. Since the calculated insertion scores are for symbols inserted to the right of the consensus symbol, the right emission insertions are on the wrong side. This can easily be corrected by noting that insertions to the left of position *i* are the same as insertions to the right of position *i*-1.

## III. Search Methods

The search algorithms will be investigated for relative performance on the problem of finding high scoring database starting positions and insertion/deletion pattern combinations. The first is a simple steepest descent algorithm that will be used to determine if more complicated genetic algorithm searches are really adding performance or just finding the obvious. If the fitness landscape is devoid of significant areas of local-optimum basins, then there is no need for an algorithm such as a genetic algorithm designed to jump out of these basins. The second two algorithms are variations on a genetic algorithm search with and without biased mutation operators. All algorithms will use the same initialization scheme to find promising starting positions. The three algorithms differ in how new trial insertion/deletion patterns are chosen based on the scores of the candidate solutions already evaluated.

### A. *Algorithm Initialization for all Search Algorithms*

The database is scanned over every possible starting position using the ungapped consensus sequence and secondary structure of the model. This is equivalent to evaluating one candidate solution at every database starting position, each with an alignment vector composed of all 1s. This will tend to give a higher score at starting locations near true starting locations since some portion of the alignment is likely to be ungapped. The portion of the alignment that is incorrect due to improper gaps should just contribute noise to the score in the same way that the entire alignment contributes noise when the starting location is nowhere near a true RNA family gene. As a result of the reduced signal (not all symbols that could be aligned are) and increased noise (from misaligned symbols), the ungapped scores will have a lower signal to noise ratio than the best alignment at each database starting position.

Candidate database starting positions are chosen from among the highest-scoring positions in the ungapped database scan. Due to the reduced signal to noise ratio, some of these starting positions are likely to not be near a true gene and expansion of the search about this database location will not be fruitful. Starting positions that are near true genes should see a score increase if the search is able to find a better alignment.

### B. *Steepest Descent*

The simple steepest descent algorithm used here evaluates a solution one step in every direction from the current solution and takes the best scoring solution among the initial solutions and the newly evaluated solutions as the next solution. This is repeated until the total number of allowed fitness evaluations is consumed. Each candidate starting position gets the same number of evaluations. After the all evaluation rounds are finished, the candidate starting positions are ranked by score for a ranked list of putative RNA genes.

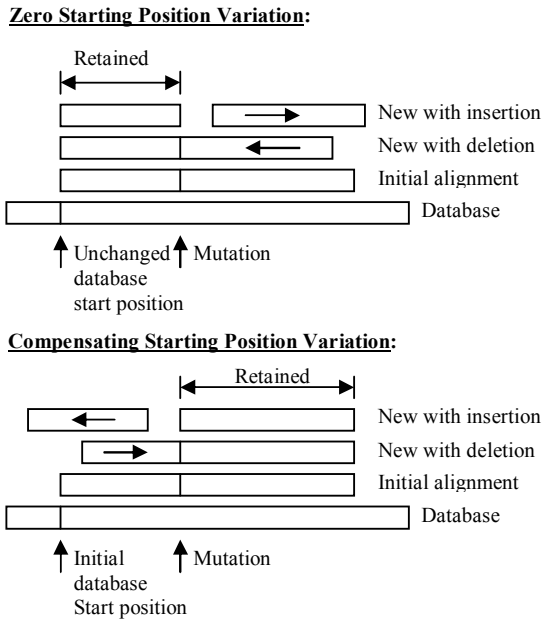


Fig. 2. Compensating and zero variations in starting position.

The search directions investigated are the insertion of an additional database symbol or the deletion of a database symbol at each consensus position (leaving all other representation vector values unchanged). These insertions or deletions can be done with or without compensation to the database starting position. The concept of compensation is shown in Figure 2. Without compensation (top half of Figure 2), insertions push the model positions to the right of the insertion point further right in the database. Deletions push the model positions to the right of the deletion point to the left in the database. The alignment between model and database to the left of the insertion or deletion point is retained. This is not helpful if the poorly aligned positions are near the left of the model. With database starting position compensation, the alignment to the right of the insertion or deletion point remains unchanged and left portion of the alignment is shifted. To make a compensating change to the database start position, an amount equal in magnitude to the alignment vector

consensus value change, but opposite in sign, must be made to the database start position.

The resulting number of fitness evaluations per round for this algorithm is $4L$, where $L$ is the number of consensus positions in the model. In practice, this number is too large to allow enough rounds to cover the total number of insertions and deletions observed in real RNA families if the total number of fitness evaluations is constrained to a value previously shown to be adequate for genetic algorithm search. In order to give the steepest descent a better chance against the GA algorithms, an approximation is used to allow more rounds and fewer evaluations per round. Instead of search over the four types of changes (insertion, deletion, insertion with compensation, and deletion with compensation) at every consensus position, the four types are tested at every $n$th position. An $n$ value of 5 is used in the results section of this work. This introduces a small reduction in the best achievable score in that it may not be possible to align up to two of the database symbols with the model, but the ability to go enough rounds to generate the needed number of total insertions and deletions is more important.

### C. *Genetic Algorithm with Unbiased Mutation*

The genetic algorithm with unbiased mutation uses a uniform distribution to choose an insertion or deletion point and then chooses to compensate the database starting point or not with one half probability of each. In each new generation the fittest individual is retained without change (elitism). Single-point crossover without mutation is applied to a random pair of individuals from top portion of the fitness ranking. In the results section, thirty five individuals per generation were used and four new individuals were produced each by crossing two individuals from the top twenty with uniform probability.

Another group of new individuals is produced by single-point mutation without crossover. In this group, the mutation takes the form of adding or subtracting 1 from a single alignment vector element. This is done with or without compensation of the database start position with equal probability. In the results section, twenty five of each generation's individuals were generated this way and the individuals to mutate were uniformly chosen from the top five fitness individuals from the previous generation.

The last group of new individuals is produced with a larger range of allowed mutation value changes. Single point mutation is still used with and without compensation. In the results section, five new individuals were generated from uniformly selected past individuals in the top five of fitness. The alignment vector changes were uniformly chosen in the range +7 to -1. The justification for this is to allow the algorithm to jump out of local minima when multiple insertions appear at a consensus position. In this case, the score may not monotonically increase as single insertions are added. An exploration of the fitness landscape in the results section indicates that this lack of monotonic behavior is likely.

### D. *Genetic Algorithm with Biased Mutation*

The unbiased-mutation genetic-algorithm search does not take into account that some mutation points are more costly in terms of insertion or deletion penalties than others and are therefore less likely to lead to an increased score. Rather than select a mutation point uniformly along the representation vector, some number $p$ of potential mutation points are drawn from a uniform distribution ($p = 20$ for the results in the next section) and the potential mutation point with the lowest insertion penalty is chosen from these $p$ potential mutation points if the mutation is positive. If the mutation is negative, then the potential mutation point with the lowest deletion penalty is chosen. In practice, positions along the consensus sequence with low insertion penalties tend to be near positions with low deletion penalties and vice versa. In other words, there tend to be brittle ranges of the sequence where both insertions and deletions are relatively common and conserved ranges where insertions and deletions are uncommon.

## IV. Comparision of Search Methods

To investigate whether there might be some value to implementing a genetic algorithm for this RNA gene search over using a simple deterministic search and whether using biased mutation further improves the search, the three algorithms are compared on a data set composed of the fourteen known U12 genes [6-8] and an assortment of randomly chosen other non-coding RNA genes from the Rfam database [9, 10]. The data set was generated by concatenating all of the U12 and non-U12 gene sequences into a single sequence of length 15880 bases. Since the regions between the U12 genes are other RNA genes, they contain stem structures that are more likely to incorrectly match the covariance model for U12 than randomly generated bases.

The first step in the search for all three search methods (steepest descent, unbiased GA, and biased GA) is an ungapped scan of the database using the covariance model single and pair matching parameters. Figure 3 shows the result of this ungapped scan of the 149 base consensus sequence against every possible 149 base window into the database sequence. From the scores in Figure 3, the 100 best scoring database start positions are chosen as initial search locations. The fourteen stars in the figure show the correct database start positions and the best possible score for each of the true U12 family members. Four of the fourteen cases can not be further improved since the database sequence has no insertions or deletions with respect to the consensus sequence. All three search methods will finish with these four solutions since none of them ever discard the best current solution. Eight of the true family members have good initial scores, but have insertions and/or deletions with respect to the model consensus sequence. Searching over alignment patterns near these database positions should bring the score up closer to the optimal score, whereas searching alignment patterns near peaks generated by noise should not increase the score much. The search procedure is thus undertaken to improve the signal to noise ratio between database positions of true family

members and database positions without true family members. Two of the true U12 family members have peaks which are very close to the noise. In particular the peak at position 446 is only the 22nd highest peak even though there are only fourteen true positives. This is a result of four different groups of deletions relative to the consensus spread out over the sequence such that there is no large ungapped subsequence for this family member.

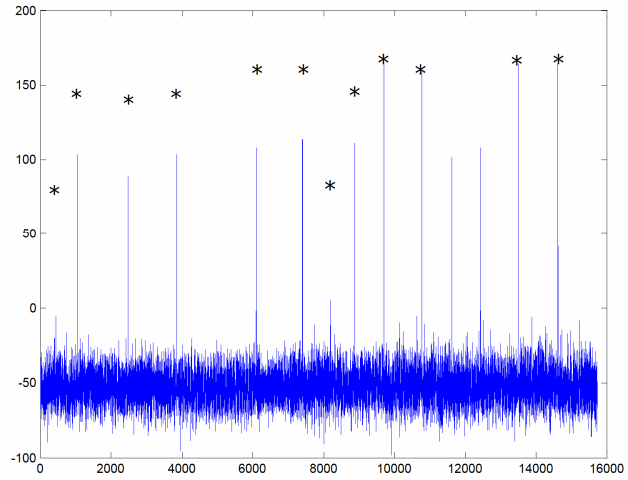


Fig. 3. Ungapped scores versus best gapped scores.

The three algorithms were then applied to the 100 initial solutions. Each algorithm was allowed 70,000 additional fitness evaluations in addition to the 100 initial fitness values. In all three cases, these were distributed as 700 evaluations starting at each of the 100 initial solutions. It is possible that more intelligent algorithms would evaluate which of the 100 searches was progressing and which were stalled and dynamically reallocate extra evaluations, but no such method was implemented in this study.

For both GAs, the 700 evaluations per position were used as 20 generations with 35 individuals per generation. For the steepest descent, six rounds were implemented with the best of 116 evaluations retained after the round. The steepest descent algorithm evaluated increasing or decreasing every fifth location in the representation vector both with and without compensation of the database start position. Using representation vector positions 5, 10, 15, ... , 145 as the test points results in 29 tested positions. Each test position had four possible changes (+1, -1, +1 with a decrease of database start by 1, and -1 with an increase in database start position by one).

Table II shows the fourteen U12 family members and the scores obtained by the three search methods after the allowed 70,000 evaluations. The Database Start Position column shows the true location of the sequence start in the database. The scores for each method are the highest score from any of the 100 solutions with a database start position in a window of eleven positions about the true start position. Four of the cases (9705, 10774, 13493, and 14615) are uninteresting since the initialization already found the best possible alignment and nothing the search algorithms did could possibly increase the score. The scores for the two GAs are averages over ten runs of each algorithm. It is clear that some of the alignments are so easily found as to not be of much interest. These include 7406 where the steepest descent found the optimal solution and both GAs found the optimal solution in ten out of ten runs. Other cases that are not particularly interesting are 1039, 3858, 8880, 11624, where steepest descent found the optimum and the GAs found the optimum most of the time. The most interesting cases are 446 and 8196, where the initialization scores (in units of bits relative to random sequence scores) are unimpressive. In both these cases steepest descent failed to improve on the initialization scores. However, both GAs improved these scores to the point that they would clearly be statistically significant even in a search of a very large database. Two other cases (6096 and 12428) also showed a rather large increase in score with the two GAs, whereas steepest descent failed to make any headway. The remaining case (2475) showed about half of the possible improvement no matter which algorithm was used. The overall conclusion is that the GAs did the best job of bringing the genes with the lowest initial and optimal scores up. Since it is these marginal cases that are of the most interest, the GAs would seem to be the preferred method, at least on this limited set of test data.

TABLE II
SEARCH METHOD COMPARISION RESULTS

| Database Start Position | Steepest Descent Score | Mean GA Score | Biased Mean GA Score | Accession Number / Nucleotide Positions |
|---|---|---|---|---|
| 446 | -0.74 | 43.68 | 50.00 | L43844.1/2-149 |
| 1039 | 146.37** | 145.66 | 144.73 | AC087420.4/142608-142466 |
| 2475 | 124.95 | 123.88 | 125.06 | AC112938.11/234142-234291 |
| 3858 | 146.37** | 143.34 | 146.37** | AL591952.9/131760-131611 |
| 6096 | 110.92 | 133.57 | 131.76 | AL669944.8/2483-2625 |
| 7406 | 159.12** | 158.12** | 159.12** | AC133939.4/22042-22191 |
| 8196 | 5.24 | 40.85 | 52.30 | AC132590.3/81080-80927 |
| 8880 | 147.13** | 147.13** | 138.13 | AL772347.6/146375-146226 |
| 9705 | 164.47* | 164.47* | 164.47* | L43843.1/2-150 |
| 10774 | 159.12* | 159.12* | 159.12* | L43846.1/332-480 |
| 11624 | 160.80** | 160.30 | 159.70 | J04119.1/2-150 |
| 12428 | 110.92 | 125.88 | 139.92 | L43845.1/358-512 |
| 13493 | 164.47* | 164.47* | 164.47* | Z93241.11/76642-76790 |
| 14615 | 164.47* | 164.47* | 164.47* | AL513366.11/57717-57871 |

*Database sequence aligns to model without gaps. No improvement possible over initial alignment.
**Search method finds the best possible alignment (steepest descent), or found best possible alignment in all ten runs (GAs). No further improvement possible.

If one concludes, as we did above, that the GA is the better algorithm, the next question is whether the added complexity of biasing the mutation operator toward brittle portions of the gene model is worthwhile. Table III shows a statistical analysis of the difference between the mean scores of the biased and unbiased GAs. Since twenty runs were used to estimate the two means for each case, there are eighteen degrees of freedom. The t-statistics associated with the difference of the means are shown. These statistics could not be calculated for the five cases where there was no variance. There are four cases where one can reject the null hypothesis that the means are the same in favor of the biased GA giving a higher score. In all other cases the difference is not statistically significant at any reasonable confidence level. Two of the four statistically significant cases are the two difficult cases of high interest. One concludes that biased mutation is potentially worthwhile. Analysis of more data and optimization of the parameter *p* (the number of potential mutation points drawn before selecting an actual mutation point) might lead to a firmer conclusion.

TABLE III
STATISTICAL SIGNIFICANCE OF BIASED SEARCH IMPROVEMENT

| Database Start Position | Biased - Unbiased Mean | t-statistic | Significance |
|---|---|---|---|
| 446 | 6.32 | 1.60 | **90%** |
| 1039 | -0.92 | -0.59 | < 90% |
| 2475 | 1.17 | 0.49 | < 90% |
| 3858 | 1.69 | 1.57 | **90%** |
| 6096 | -1.81 | -0.36 | < 90% |
| 7406 | 0 | - | - |
| 8196 | 11.46 | 1.82 | **95%** |
| 8880 | -1.43 | -0.97 | < 90% |
| 9705 | 0 | - | - |
| 10774 | 0 | - | - |
| 11624 | -0.60 | -0.97 | < 90% |
| 12428 | 14.04 | 3.39 | **95%** |
| 13493 | 0 | - | - |
| 14615 | 0 | - | - |

With 18 degrees of freedom, the one-tailed t-statistic critical values for 90% significance and 95% significance are 1.33 and 1.73 respectively. The null hypothesis is that the means are equal and the alternative is that the biased GA has a higher mean.

Finally, we investigate why the GA methods seem to outperform the steepest descent algorithm. Figure 4 shows a very small portion of the fitness landscape that is evaluated by the search algorithms. The figure contains nine graphs showing a search region around an alignment representation vector composed off all ones with the exception of vector position 60. The plot in the top row and middle column shows deviations from a base ungapped alignment near the database positions of one of the U12 genes (the U12 gene at 12428). The plot in the upper left is similar except the deviations are from a base with no gaps except a deletion of position 60. The deviations take the form of scoring the alignment for all position 30 alignment vector values in the range 0 to 9 and all database start positions in the range 12425 to 12434. The U12 gene starting a database position 12428 has six inserted bases after consensus position 34 and no other insertions or deletions. The best possible alignment if we are only allowed to change alignment vector positions 30 and 60 is to have the values 7 and 1 respectively (resulting in only database symbols 31, 32, 33, and 34 being misaligned). This can been seen as the dark square at *start deviation* = 0, and *representation value 30* = 7 in the top middle plot. A horizontal band a *start deviation* = 0 in each plot shows that very good scores are generally found if the correct start position in the database has been found. However, the slope down to the optimum in each plot is not monotonic even once the correct database start position is found. It is also apparent that there are numerous local minima all over each of the plots. The need for some sort of algorithm that can get out of local minima is clear.

## V. Conclusions

A continuation of the investigation started in [3] as to the usefulness of genetic algorithms in accelerated covariance-model-based non-coding RNA gene search was undertaken. The genetic algorithm was compared with a simple deterministic algorithm to see if the additional complexity of the GA was generating any better performance. This comparison was lacking in the previous study. It appears that the GA (or some type of non-deterministic algorithm) is justified. The possibility of biasing the search toward alignments with insertions and deletions was also investigated. Although the biased version studies here does seem to outperform the unbiased version in a statistical sense, it is not yet clear that the performance improvement is meaningful overall. Tests on much larger data sets and parameter tuning will be necessary if the conclusive statement about the worth of this biased approach is to be made.

It should also be noted that the results of the unbiased GA are not the same as in [3]. The reason for this is that the original paper had one large population with diverse database starting positions. A large elite (30 members) was maintained with some complicated elite selection methods in order to maintain this diversity. This only partially worked. The U12 genes at 446 and 8196 were not present in the final population. Maintaining many separate populations has proved to work much better, so that is the method used in this and likely all future work.

## 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] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*, Cambridge University Press, 1998.
[3] S. Smith, "Covariance Searches for ncRNA Gene Finding," *IEEE Symp. on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 320-326, 2006.
[4] J. Yadgari, A. Amir, and R. Unger, "Genetic Threading," Constraints 6, pp. 271-292, 2001.
[5] Infernal Users Guide, http://www.genetics.wustl.edu/eddy/infernal/.
[6] G. Shukla and R. Padgett, "Conservation of Functional Features of U6atac and U12 snRNAs Between Vertebrates and Higher Plants," *RNA*, 5, pp. 525-538, 1999.
[7] W. Tam and J. Steitz, "Pre-mRNA Splicing: The Discovery of a New Spliceosome Doubles the Challenge," *Trends in Biochemical Science*, 22, pp. 132-137, 1997.
[8] L. Otake, P. Scamborova, and J. Steitz, "The Divergent U12-type Spliceosome is Required for Pre-mRNA Splicing and is Essential for Development in Drosophila," *Mol Cell*, 9, pp. 439-446, 2002.
[9] Rfam website, http://rfam.janelia.edu.
[10] 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.

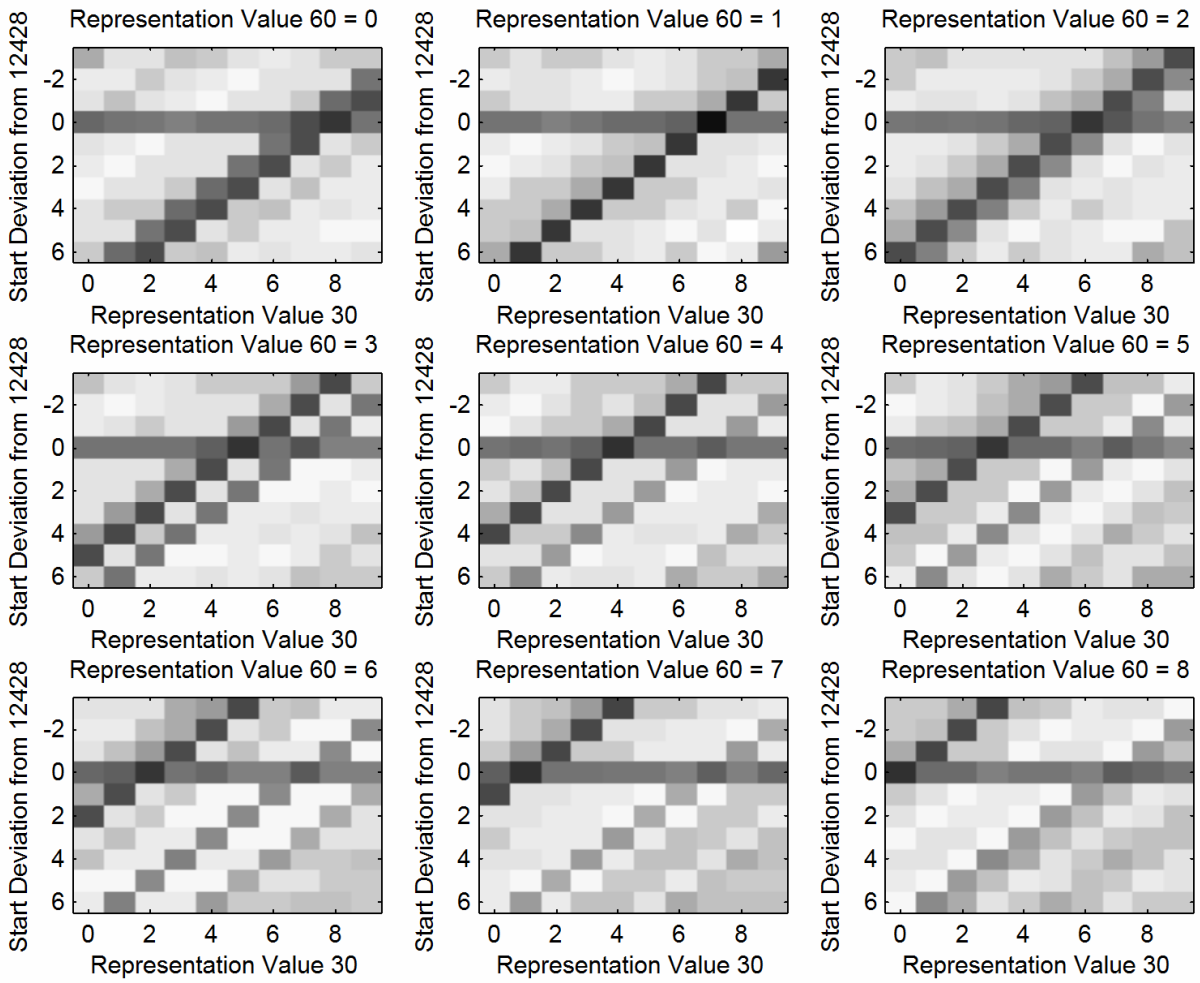


Fig. 4. Example fitness landscape.