INCOMPLETE GENE STRUCTURE PREDICTION WITH ALMOST 100% SPECIFICITY

A Thesis

by

SEE LOONG CHIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2003

Major Subject: Computer Science

INCOMPLETE GENE STRUCTURE PREDICTION WITH ALMOST 100% SPECIFICITY

A Thesis

by

SEE LOONG CHIN

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Sing-Hoi Sze (Chair of Committee)	Thomas Ioerger (Member)
Jin Xiong (Member)	Valerie E. Taylor (Head of Department)

December 2003

Major Subject: Computer Science

ABSTRACT

Incomplete Gene Structure Prediction with Almost 100%
Specificity. (December 2003)
See Loong Chin, B.S., Purdue University
Chair of Advisory Committee: Dr. Sing-Hoi Sze

The goals of gene prediction using computational approaches are to determine gene location and the corresponding functionality of the coding region. A subset of gene prediction is the gene structure prediction problem, which is to define the exon-intron boundaries of a gene. Gene prediction follows two general approaches: statistical patterns identification and sequence similarity comparison. Similarity based approaches have gained increasing popularity with the recent vast increase in genomic data in GenBank.

The proposed gene prediction algorithm is a similarity based algorithm which capitalizes on the fact that similar sequences bear similar functions. The proposed algorithm, like most other similarity based algorithms, is based on dynamic programming. Given a genomic DNA, $X = x_1 \cdots x_n$ and a closely related cDNA, $Y = y_1 \cdots y_n$, these sequences are aligned with matching pairs stored in a data set. These indexes of matching sets contain a large jumble of all matching pairs, with a lot of cross over indexes. Dynamic programming alignment is again used to retrieve the longest common non-crossing subsequence from the collection of matching fragments in the data set.

This algorithm was implemented in Java on the Unix platform. Statistical comparisons were made against other software programs in the field. Statistical evaluation at both the DNA and exonic level were made against Est2genome, Sim4, Spidey, and

Fgenesh-C. The proposed gene structure prediction algorithm, by far, has the best performance in the specificity category. The resulting specificity was greater than 98%. The proposed algorithm, also has on par results in terms of sensitivity and correlation coefficient. The goal of developing an algorithm to predict exonic regions with a very high level of correctness was achieved.

To my family for their continuous support over all these years.

ACKNOWLEDGMENTS

I would like to express my thanks and gratitude to Dr. Sze, Dr. Ioerger, and Dr. Xiong for their guidance and advice. I would especially like to thank Dr. Sze for his encouragements and patience with me. I would also like to thank Dr. Childs who made all this possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A. Background

The goals of gene prediction using computational approach are to determine gene locations and the corresponding functionality of the coding region. A subset of gene prediction is the gene structure prediction problem, which is to define the exon-intron boundary of a genomic sequence. Gene prediction follows two general approaches: statistical patterns identification and sequence similarity comparison. Coding exonic regions of a genomic sequence display certain characteristic attributes, such as codon usage, hexamer measure, and amino acid usage measure [1]. These statistical measures are collected and categorized from a known organism. A model trained with statistical attributes from one organism can be used to predict the coding regions of homologous species [2], [3], [4] and [5].

1. Statistical Approach

Burge and Karlin [2] proposed a general probabilistic model based on the Hidden Markov Model approach. Transcriptional, translational, and splicing signals are statistical information used to train this model. Unlike most other approaches which rely solely on independent statistical attributes, Burge and Karlin [2] introduced a Maximal Dependence Decomposition method to keep track of dependencies between signals. GENSCAN is the computer program implementation of this model.

Gelfand [3] introduced a method where the local prediction of splicing sites are combined with global prediction to offer several possible best results. The predicted

---

The journal model is *IEEE Transactions on Automatic Control*.

results can be further refined with user input if certain information such as the number of exons is known.

Uberbacher and Mural [4] used a combination of multiple sensing algorithm with a neural network to find the best coding region. Each of the seven sensor algorithms looks for specific attributes such as frame bias, regularity of 6-tuple words, and word commonality. These results are passed through a neural network, that has been trained with a homologous data set, to return a closest fit result.

Another exon prediction method was proposed by Solovyev *et al.* [5]. An algorithmic prediction of donor and acceptor sites provides splice site information. Splice site results are combined with triplet frequencies to return prediction of exon-intron regions.

All these methods are statistical approaches that use previous known data to predict future results. So, the obvious drawback of this approach is that the gene to be predicted must have similar characteristics of the training set for good results.

2. Sequence Similarity Approach

With the advent of GenBank and the increasing availability of genomic data, the sequence similarity approach has gained increasing feasibility. Similarity based approaches capitalize on the fact that similar nucleotide sequences have similar functionality, and the splicing sites have splicing constraints. In the sequence similarity approach, a cDNA sequence is used to identify the exonic regions of a genomic DNA from a related organism. This method applied to closely related homologous organisms can yield very good prediction of genomic exons.

Many of the sequence similarity based approaches implement some variant of the global alignment algorithm by Needleman and Wunsch [6], or the local alignment algorithm by Smith and Waterman [7]. Global alignment algorithms are useful for

comparing two sequences that are similar throughout the entire sequence, whereas the local alignment algorithms are more suitable for comparing sequences with locally similar regions. Both global alignment algorithm and local alignment algorithm are dynamic programming algorithms. Dynamic programming is useful in circumstances where the problem can be partitioned into non-independent subproblems which are stored in a table to be reused.

Given any two sequences $X = x_1 \cdots x_n$ with length m, and $Y = y_1 \cdots y_n$ with length n, the global alignment algorithm in equation 1.1 will return an optimal solution. The maximum score $S_{i,j}$ is selected from possible matches, mismatches, and indel cases. Matching pairs are usually assigned positive scores with mismatches/indels assigned zero or negative scores. Compute the maximum score, $S_{i,j}$ for each (i,j) position. $S_{m,n}$ is the optimal score between two sequences using the global alignment algorithm. The running time is proportional to m x n. The optimal aligned sequence can be retrieved by storing the pointers, for each (i,j), of the location where the maximum $S_{i,j}$ was obtained. These pointers are stored in a separate table. Backtracking from location (m,n), returns the maximum aligned sequence.

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + \delta_{match} & if\ a_i\ =\ b_j \\ S_{i-1,j-1} + \delta_{mismatch} & if\ a_i\ \neq\ b_j \\ S_{i-1,j} + \delta_{indel} & \\ S_{i,j-1} + \delta_{indel} & \end{cases} \tag{1.1}$$

Smith and Waterman proposed modifications to the global alignment algorithm to perform matching of subsequences. Local alignment, equation 1.2, allows matching sequences to begin and end anywhere. The 0 case permits a new starting point at

any position (i,j) when all the other cases result in a negative score. The optimal local alignment now starts at the maximum value of $S_{i,j}$.

$$S_{i,j} = max \begin{cases} 0 \\ S_{i-1,j-1} + \delta_{match} & if\ a_i\ =\ b_j \\ S_{i-1,j-1} + \delta_{mismatch} & if\ a_i\ \neq\ b_j \\ S_{i-1,j} + \delta_{indel} \\ S_{i,j-1} + \delta_{indel} \end{cases} \tag{1.2}$$

Gap3 by Huang and Chao [8] is a modified global alignment algorithm used for comparing sequences with intermittent similarities. Gap3 is capable of finding regions of low similarities at the expense of compute time.

Est2genome uses a combination of local alignment and global alignment to provide an optimal solution [9]. An initial alignment with Smith-Waterman algorithm is used to find local subsequences. These subsequences are extracted and aligned with the Needleman-Wunsch algorithm, if the product of the subsequence length is less than the area parameter. If the subsequences are too long, then the EST is recursively split until the extracted subsequences meets the area parameter threshold.

Sim4 [10] uses an algorithmic approach that is very similar to BLAST [11], a local alignment search tool. An initial pass searches for local exact matches that are then extended to achieve a maximal scoring gap-free segment. These highly matching sections are further extended to neighboring unmatched fragments using greedy algorithms.

Sze *et al.* [12] utilized the Las Vegas algorithm to maximize the specificity, and correctness of the predicted regions. Las Vegas algorithms only return correct re-

sults and would not return any result otherwise. Thus, only correct exonic region predictions are returned.

CHAPTER II

PROCEDURE

A. Problem Statement

Given a section of the genomic DNA, find the intron-exon regions when the homologous cDNA or EST is provided. The accuracy of the prediction is largely dependent on the evolutionary distance between the genomic DNA and the cDNA. For closely evolutionary related organisms, the exonic regions of an unknown organism can be predicted using known cDNA of close relations. DNA's are composed of exons, small coding regions, separated by vast regions of non coding introns. Figure 1 shows the DNA structure with exons and introns.

This biological problem can be represented as the comparison of two strings with intermittent similar regions. These strings have short highly matching regions that are separated by long dissimilar regions. Matching of two strings is a problem that can be optimally solved using dynamic programming. Specifically, Longest Common Subsequence(LCS), is a dynamic algorithm used to compare two similar strings.

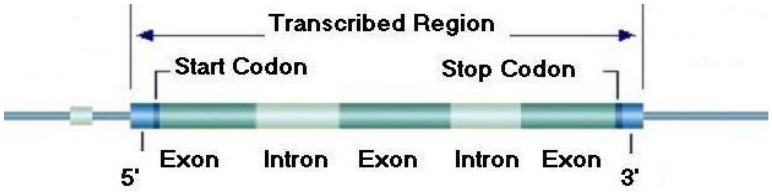

Fig. 1. DNA Structure

B. Longest Common Subsequence Algorithm

The basic building block of this gene structure prediction algorithm is based on the longest common subsequence (LCS) algorithm. LCS is a dynamic programming algorithm that returns the longest common matching base pairs of two strings. Dynamic programming algorithms have 4 fundamental steps [13]:

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution in a bottom-up fashion.
4. Construct an optimal solution from computed information.

The proof of step 1 of the dynamic programming in characterizing the optimal solution is in [13]. The next step, step 2, of the dynamic programming defining the value of the optimal solution is satisfied with the following recursive formula, equation 2.1, from [13].

$$C_{i,j} = \begin{cases} 0 & if\ i\ =\ 0\ \ or\ \ j\ =\ 0 \\ C_{i-1,j-1} + 1 & if\ i,j\ >\ 0\ \ and\ \ x_i\ = y_j \\ max(C_{i,j-1}, C_{i-1,j}) & if\ i,j\ >\ 0\ \ and\ \ x_i\ \neq\ y_j \end{cases} \tag{2.1}$$

Step 3 of the dynamic programming constructs a table for storing the values of all possible combinations of base pair matches. The following code fragment implements step 3. For a matching base pair, add one to the diagonal top-left value and let this be the value of the current position. For non matching base pairs, take the largest value of the horizontal or vertical positions.

```
for (int i=1; i<=m; i++) {
      for (int j=1; j<=n; j++) {
        if (x[i-1] == y[j-1]) {
          C[i][j] = C[i-1][j-1]+1;
        }
        else if (C[i-1][j] >= C[i][j-1]) {
          C[i][j] = C[i-1][j];
        }
        else {
          C[i][j] = C[i][j-1];
        }
      }
    }
```

The final step 4 constructs the longest matching base pairs; this is accomplished with the following while loop. This step starts from the bottom right diagonal of the table and walks up to the top left of the table taking the longest possible route. The stop condition for this loop is the index of either i or j equals 0. This is the initial index of the strings. The algorithm implemented selects the "up" step over the "left" step for cases with equivalent values. So, decrementing the i^{th} index along the x_i string is preferred.

```
 while(1) {
    if ((i==0) || (j==0)) {
      break;
    }
    else if ((C[i-1][j] < C[i][j]) && (C[i][j] == C[i-1][j-1]+1)) {
      System.out.print(C[i][j]+"diagonal ");
       i--; j--;
     }
    else if ((C[i-1][j] >= C[i][j-1]) && (C[i][j] == C[i-1][j])) {
      System.out.print(C[i][j]+"up ");
      i--;
    }
    else {
      System.out.print(C[i][j]+"left ");
      j--;
    }
} // end while
```

C. Gene Structure Prediction Algorithm

The following algorithmic approach will be used to achieve an exonic prediction with correctness close to one hundred percent. Given a genomic DNA, $X = x_1 \cdots x_n$ and a closely related cDNA, $Y = y_1 \cdots y_n$; choose a fragment of size k from X and Y. Compare all combinations of X and Y k-mers, using the longest common subsequence algorithm, LCS. This step can be implemented using nested for loops and is illustrated in figure 2.

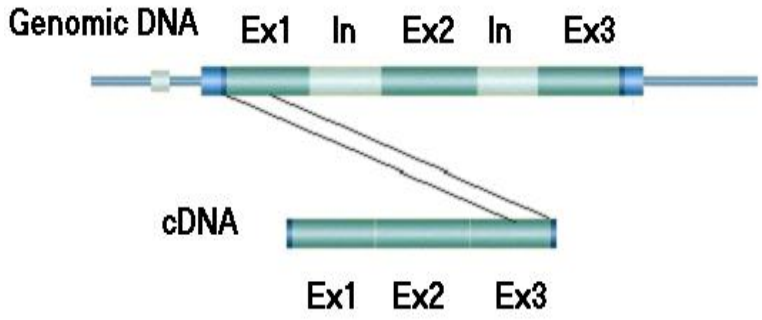

Fig. 2. Alignment of All X vs. Y Combinations

```
For each k-mer in Genomic DNA  {
     For each k-mer in cDNA  {
          Find LCS
     }
}
```

The indexes of k-mers matching pairs above a similarity threshold score, d, are held in a data set. These indexes of matching sets contain a large jumble of all matching pairs, with a lot of cross over indexes. The initial set of indexes with cross over indexes is shown in figure 3. Again, apply LCS to retrieve the longest common non-crossing subsequence from the collection of matching fragments in the data set. These steps will be applied to both DNA and amino acid sequences. For the amino acid sequence comparison, the DNA and cDNA codons will be converted to an amino acid before applying LCS. For each DNA k-mer, convert to three corresponding amino acid sequence fragment, taking into account the three frame shifts. Keep the highest of the three frame shift index pairs surpassing the threshold score in the data set.

The preliminary predictions had many small gaps in the sequences. This gap

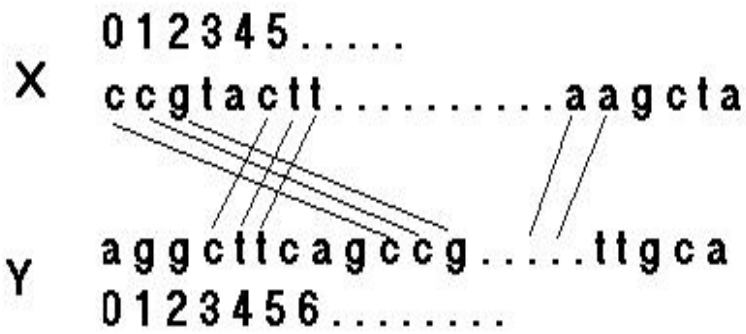

Fig. 3. Retrieval of Longest Non-Crossing Subsequence

sizes were usually between 2-4 base pairs apart. Three different methods were tried in an attempt to alleviate this small gap size problem. The initial technique was to select the entire indexes in a word if the threshold was met. For example, select the entire 20 indexes for a 20-mer if the matching base pairs exceed a threshold of 15. The second method checks the indexes for small gaps and linking the fragments separated by small gaps. The gap threshold, t, is provided by the user. The third method combines the first two approaches. Take the whole array index for matching word fragments and link any remaining gaps. Test cases showed that the first scheme provided the best prediction results. Figure 4 illustrates this step of the algorithm.

At this stage of the algorithm, continuous exonic regions in the genomic DNA have been predicted. The exon-intron boundary delineation can be refined by incorporating acceptor and donor sites information. Observation of the data showed a trend where most of the false predictions were found at the end of an exon and the beginning of an intron. The predicted regions did not stop cleanly at the end of the exon region, but over predicted around 10-20 base pairs into the intron region. An exon/intron boundary pattern matching was introduced to solve this problem. All

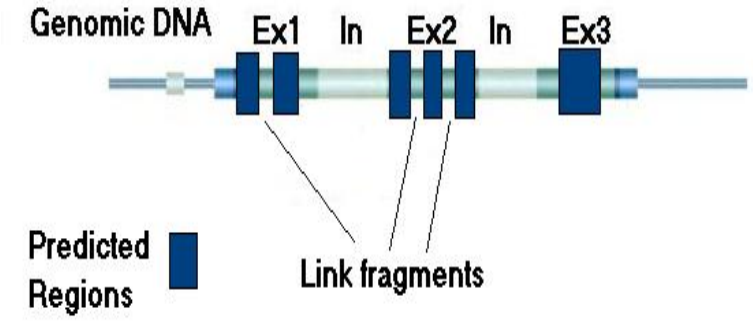

Fig. 4. Eliminating Gaps in Preliminary Prediction

the sub sequences were checked for the "gt" exon/intron boundary pattern. Start at the last index of a subsequence and walk to the front of the sequence looking for the residue "t". If found, then check if the next immediate residue is "g". If found, then remove all residues from "gt" until the end of the sequence. The user has the option to select how many residues starting from the back of the index to test for. The implemented boundary recognition steps removes the very first acceptor/donor sites found. Test cases have shown that removing from the very last acceptor/donor sites, increases the specificity at the expense of sensitivity. This step of the algorithm is illustrated in figure 5. These exons are furthered filtered by dropping exonic regions below a threshold size provided by the user.

Comparative evaluations were made against programs solving similar problems such as Sim4, Est2genome, Spidey, and Fgenesh-C. Est2genome is a dynamic programming algorithm providing an optimal solution. Sim4 and Spidey are based on the heuristic BLAST approach, using local alignment algorithms to find locally similar regions. These local regions are extended until the similarity score can no longer be improved. Fgenesh-C is a statistical based method using Hidden Markov Model

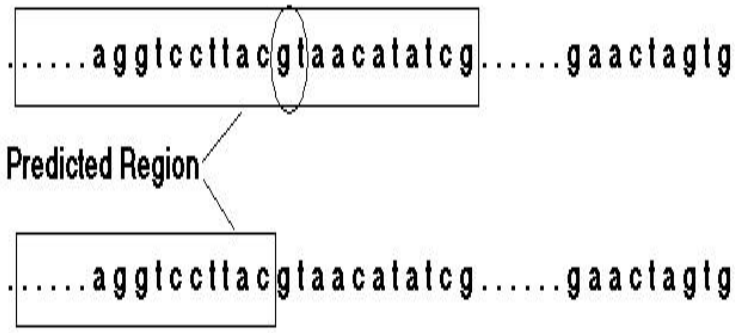

Fig. 5. Refining Boundary Location with Acceptor/Donor Patterns

with cDNA/EST as inputs. Both Sim4 and Est2genome have Unix versions that can be downloaded. Sim4, Est2genome, and my gene structure prediction algorithm were evaluated on 500 test cases. These test cases were grouped by similarity between the cDNA and the genomic DNA. The organism evolutionary distances range from identical (human cDNA vs. human genomic DNA) to highly dissimilar (bacteria cDNA vs. human genomic DNA). The web based programs (Fgenesh-C, and Spidey) were performed on a small sample set of 30 test cases.

D. Analysis of Results

The performance parameters in measuring how well the predicted residues match the actual residues utilize the method introduced by Burset and Guigo [14]. The residues are false positive (wrong hit), false negative (missed hit), true positive (correct hit), and true negative (correct miss). Figure 6 shows these regions of incorrect and correct matches. The nucleotide level accuracy equations, 2.2 to 2.5, are used to calculate the performance parameters. Sensitivity is a ratio of the number of correctly predicted exons over the number of actual existing exons. Sensitivity gives an indication of

how well the program finds the exonic residues. Specificity is a ratio of the number of correctly predicted residues over the number of predicted exonic residues. This parameter shows the confidence level of predicted values. In other words, specificity defines how correct the prediction is. Correlation coefficient provides the degree of linear relationship between two variables with negative one meaning a negative relationship, and positive one meaning a positive relationship. A value of zero indicates no relationship.

Nucleotide Level Accuracy Equations

TP - True Positive FP - False Positive

TN - True Negative FN - False Negative

Sn - Sensitivity Sp - Specificity

AC - Approximate Correlation

CC - Coefficient Correlation

$$Sn = \frac{TP}{TP + FN} \tag{2.2}$$

$$Sp = \frac{TP}{TP + FP} \tag{2.3}$$

$$AC = 0.5 * (\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN}) - 1 \tag{2.4}$$

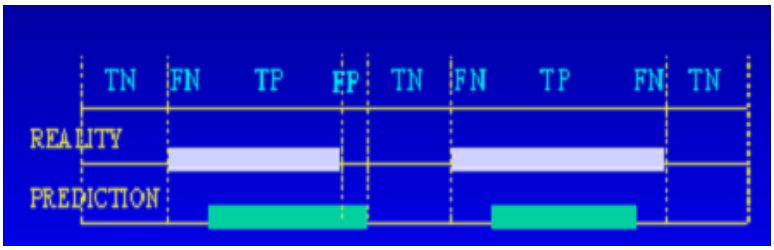

Fig. 6. Nucleotide Level Accuracy

$$CC = \frac{(TP * TN) - (FN * FP)}{(TP + FN) * (TN + FP) * (TP + FP) * (TN + FN)} \tag{2.5}$$

Results will also be presented in terms of sensitivity, specificity, and overall correlation of the predicted against true exonic regions. True exons are exonic regions where the predicted exonic regions match the actual exonic regions. Exon level sensitivity indicates how well the exonic regions were predicted. The exon level specificity indicates how correct the exonic regions were predicted. Correlation coefficient in the exonic level is represented by the average of exon level sensitivity and specificity. Missing exons are actual exons not predicted at all. Wrong exons are exons that are incorrectly predicted. The exon level accuracy equations are provided in equations 2.6 to 2.10 below.

Exon Level Accuracy Equations

TE - True Exon AE - Actual Exon PE - Predicted Exon

ESn - Exon Sensitivity ESp - Exon Specificity

ME - Missing Exons WE - Wrong Exons

$$ESn = \frac{TE}{AE} \tag{2.6}$$

$$ESp = \frac{TE}{PE} \tag{2.7}$$

$$\frac{ESn + ESp}{2} \tag{2.8}$$

$$ME = \frac{AEnotoverlappedbyPE}{AE} \tag{2.9}$$

$$WE = \frac{PEnotoverlappedbyTE}{PE} \tag{2.10}$$

E. Software Implementation

This algorithm was implemented using the Java programming language. Java was selected because it provided good modular programming support with many built in classes. Another Java advantage is that it is platform independent. All the code was developed in the Unix environment using the Unix Java compiler. A flow chart of the entire DNA version of the Java program broken down my modules is illustrated in figure 7. The main module is "testlcs.java". This module reads the input strings (genomic DNA and cDNA), and all user set parameters (word size, match threshold, gap threshold, exon/intron boundary cleanup threshold, and minimum word fragment threshold). "testlcs.java" calls all other modules for LCS analysis and accuracy calculations. The second module, "lcs.java", performs the alignment of all

combination of the DNA word fragments against the cDNA. The matching indexes are added to vectors of the object created by "match.java". "match.java" serves as a data storage module. "lcs2.java" take the indexes of all the matching fragments and finds the maximum non-crossing subsequence. "lcs2.java" also links word fragments separated by small gaps together, cleans up the exon/intron boundary region, and remove any fragments less than the required minimum size. "cc.java" does all the data manipulation and analysis required to report statistical results.

The amino acid version of the program is very similar to the DNA version of the program. The main difference is that "lcs.java" of the amino acid version calls the nucleotide to amino acid converter, "translator.java". Each genomic DNA fragment is converted to its three equivalent frame-shifted amino acid sequences. The three possible amino acids are aligned with the cDNA amino acid keeping the longest alignment. The indexes of the best alignment are kept as nucleotide indexes. This approach resulted in minimal changes to the other modules.

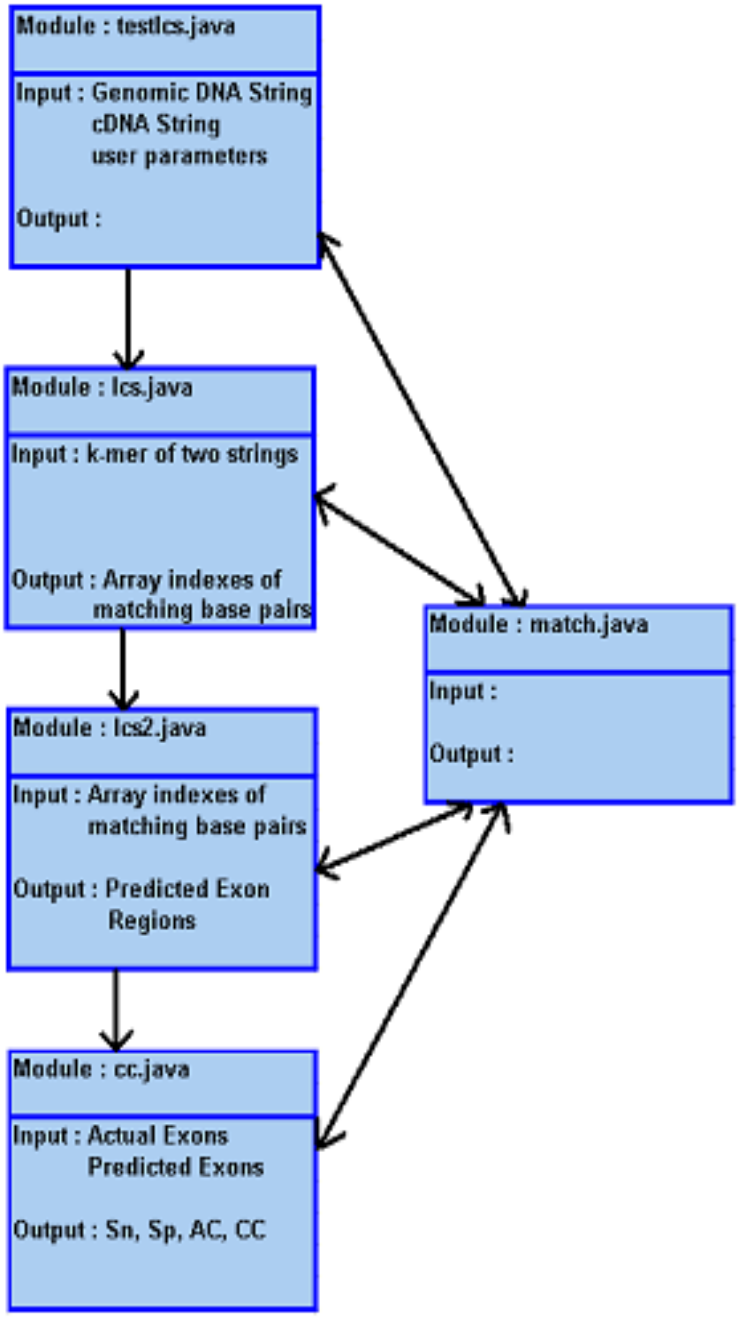

Fig. 7. Software Flowchart

CHAPTER III

DISCUSSION

A. Results

This gene structure prediction program was evaluated against other similar based programs. Sim4, Est2genome, and Spidey are similarity based programs. Fgenesh-C is a commercial program that uses the Hidden Markov Model statistical approach. The average results were collected from running approximately 500 DNA-cDNA data sets from Gelfand *et al.* [15]. The data sets shown in table I are grouped according to evolutionary distance. The evolutionary distance between cDNA and DNA increases from group 0 to group 3. Both the Sim4 and est2genome programs have copies that could be downloaded to the local Unix machines. Spidey and Fgenesh-C are web based. Only a small subset of the data set were evaluated on Spidey and Fgenesh-C. Furthermore, the commercial software Fgenesh-C only allows 10 free runs per day.

Table I. Data Set Classification

Group	DNA	cDNA
0	Human	Human
1	Human	Mammalian
2	Human	Vertebrae
3	Human	Others

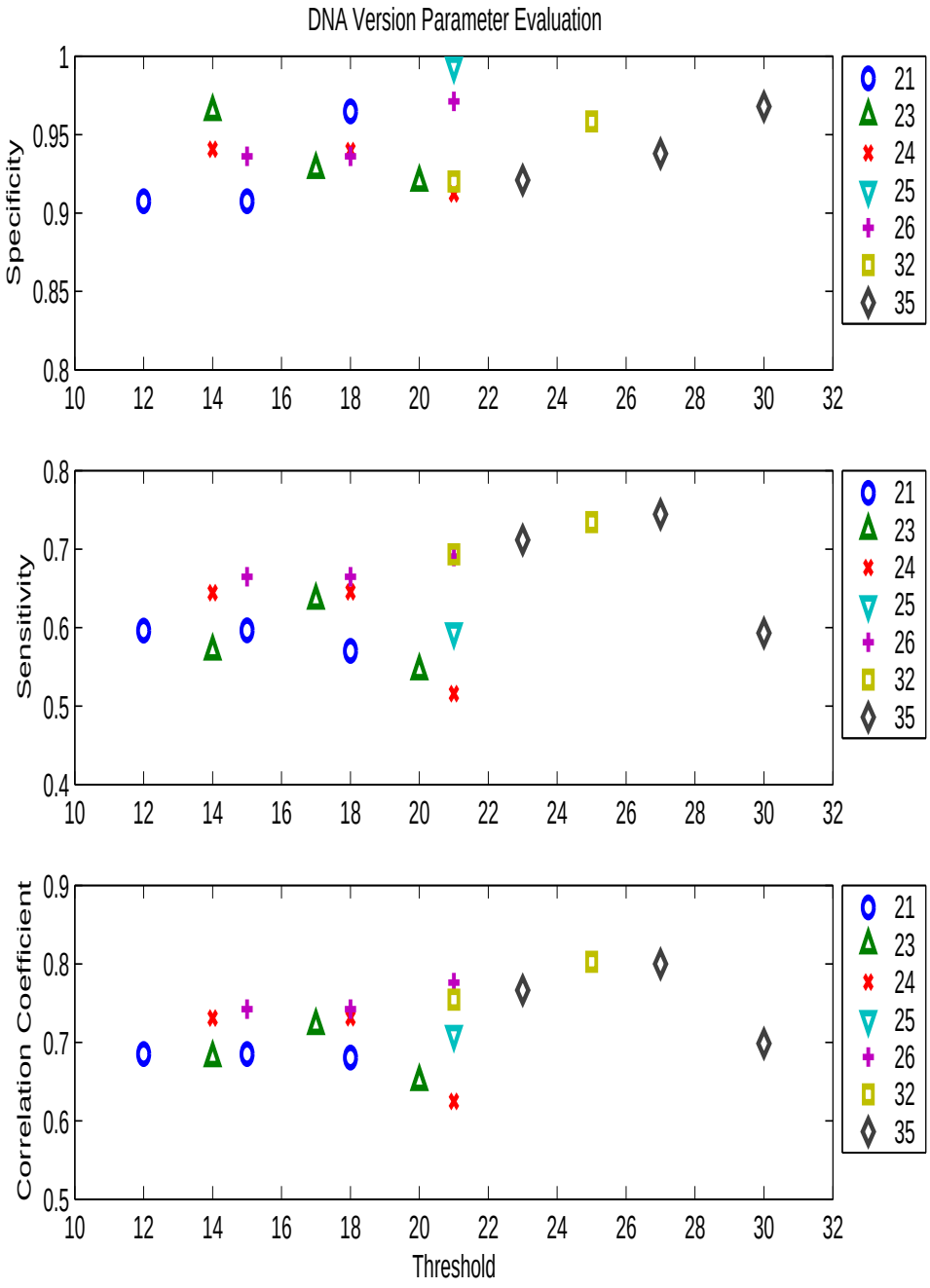

Fig. 8. DNA Version Parameter Evaluation

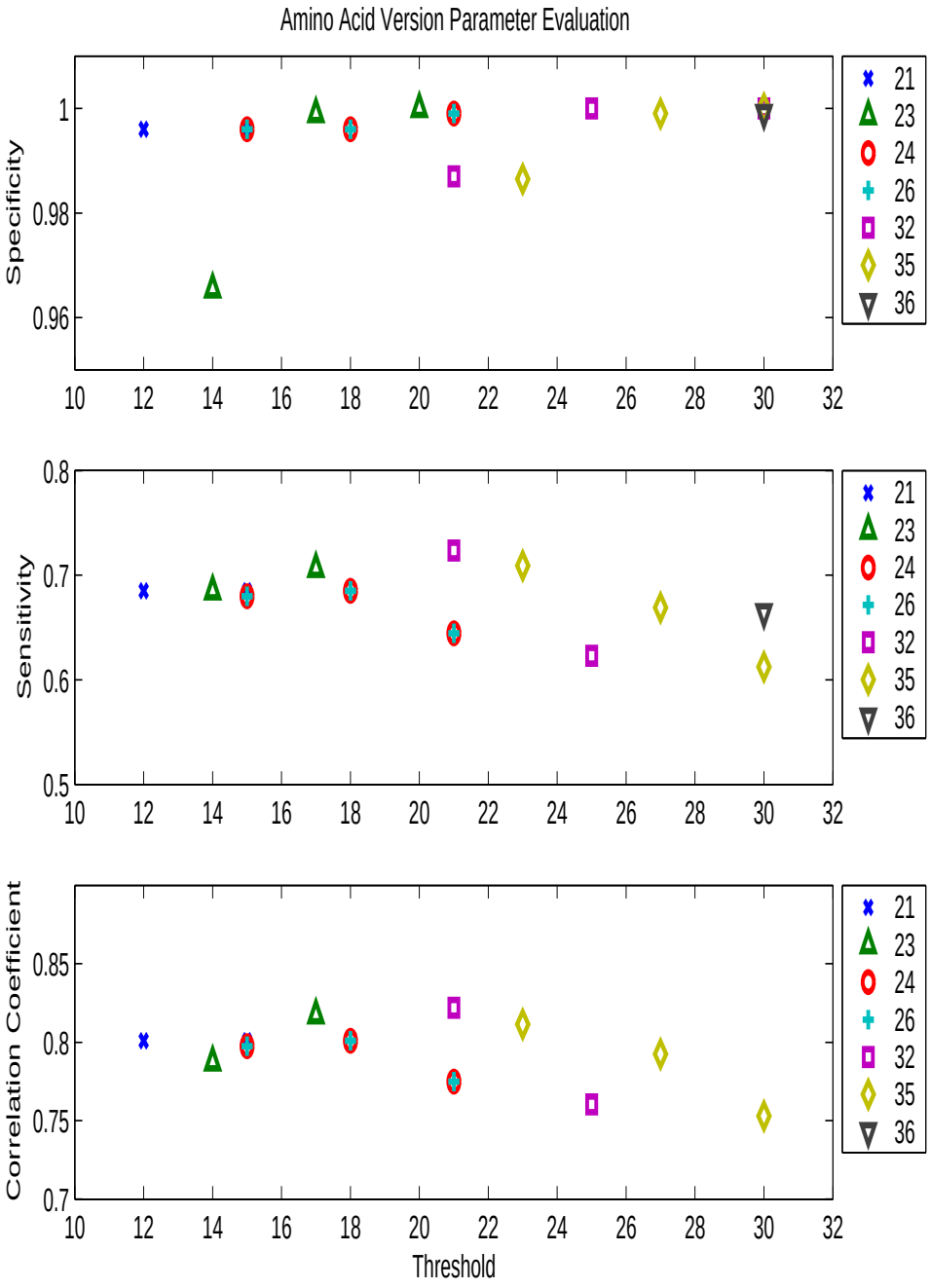

Fig. 9. Amino Acid Version Parameter Evaluation

B. Initial Evaluation

Program performance is dependent on the word size, fragment threshold, gap link threshold, boundary pattern search size, and the minimum fragment size. Word size is the initial word fragment size used to find all possible matches. The fragment threshold is the threshold that must be met in order for the returned word matches to be stored in the matching vector. Large differences between word size and fragment threshold increase sensitivity but decrease specificity. Fragments separated by gap size less than the gap size threshold are linked together. Large fragment threshold size will increase specificity but decrease sensitivity. The boundary pattern search size is the number of nucleotides that will be searched for acceptor donor patterns from the end of the fragments. Any fragment less than the minimum fragment size will be dropped. Dropping small fragment sizes provided a big contribution in increasing specificity with only a small hit in sensitivity. Initial evaluations were conducted on a small subset to zone in on the parameters that will return optimal performance. Figure 8 shows the effect of word size and fragment threshold on specificity, sensitivity and correlation coefficient for the straight DNA version of the program. There is an optimal threshold size for each word size. Cases with word size of 21, 23, 24, 32 and 35 clearly show an increase in performance in terms of sensitivity up to a certain threshold value before decreasing. Word size of 25 with fragment threshold set at 21 returned the best specificity. Word size of 32 with fragment threshold of 25 returned the best sensitivity. These results were used to choose word size of 32 and 25 with corresponding fragment threshold of 25 and 21.

Figure 9 illustrates the results obtained with varying word size and fragment threshold for the amino acid version of the program. The amino acid version shows similar trend as the straight DNA version with increasing performance up to a certain

threshold. From these preliminary runs, word size of 36 and 24 with corresponding fragment threshold of 30 and 18 were chosen for the full data set runs.

C. Protein vs. DNA Only Versions

A combination of the gene only version and the translated amino acid version results are shown in figure 10. The runs with prefixes of "P" are results from the gene to amino acid translated version. The runs with prefixes of "G" are straight genomic DNA against cDNA alignments without any form of translation. The first number following the prefix indicates the word size, with the second number indicating fragment threshold. So, "P-24-18" are results from the translated amino acid version with word size parameters of 24 and word fragment threshold of 18. All the translated amino acid runs has gap link threshold of 4, boundary pattern search size of 15, and minimum fragment size of 20. The straight gene version, on the other hand, has gap link threshold of 5, boundary pattern search size of 15, and minimum fragment size of 35. The protein version consistently provided a higher specificity than the DNA only version. This is especially true in the "P-36-30" run which far surpasses the other runs with specificity that is very close to 1. The large disparity in specificity between "P-36-30" and "P-24-18" underlines the importance of choosing the correct word and fragment threshold parameters. Sensitivity and correlation coefficient show very similar trends with decreasing performance as the distance between the genomic DNA and cDNA increases.

D. Benchmark Comparisons

This gene structure prediction program was evaluated against other similar programs such as Sim4, Est2genome, Spidey, and Fgenesh-C. The tabulated results are pro-

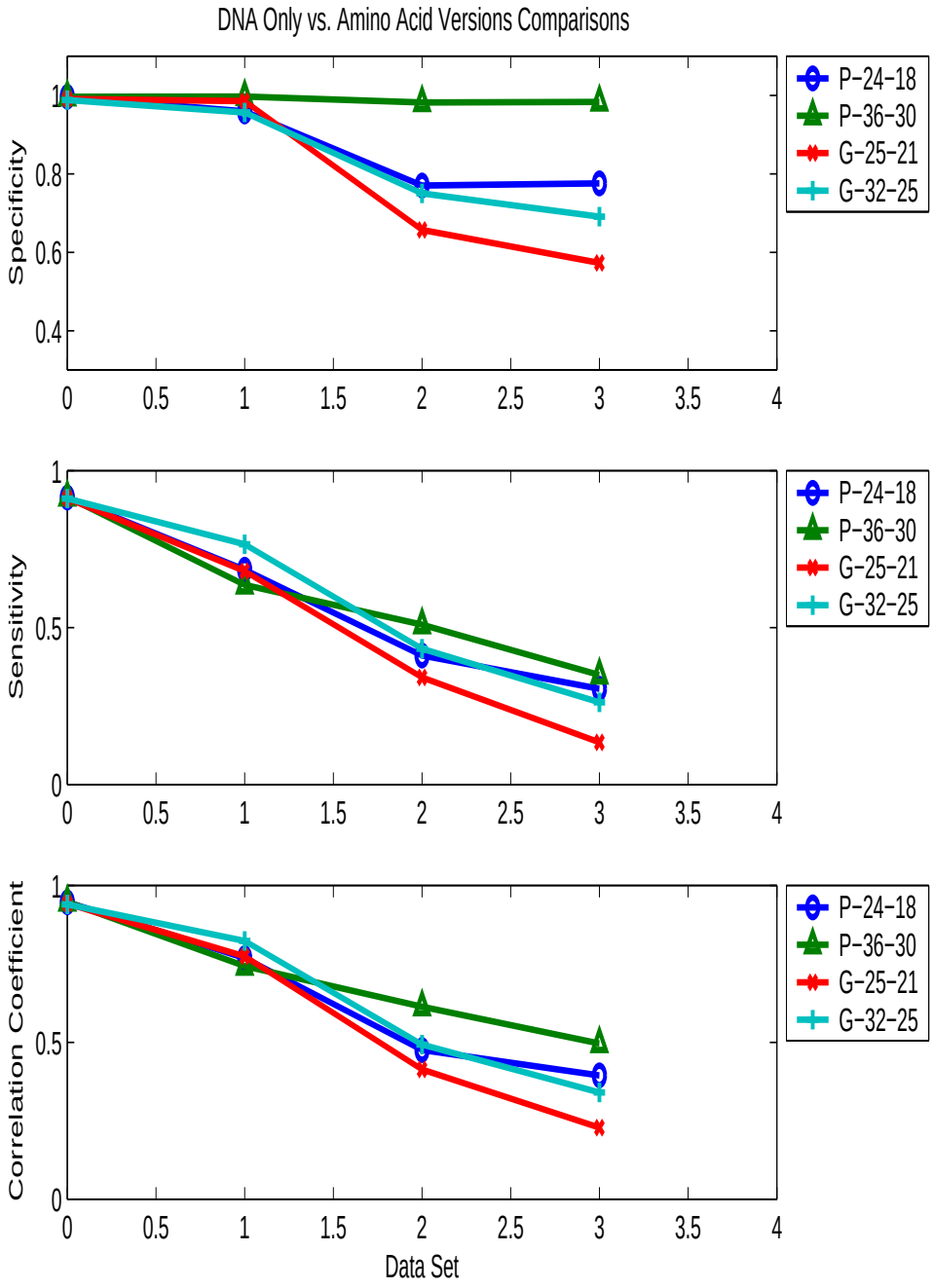

Fig. 10. DNA Only vs. Amino Acid Versions Comparisons

seq0	Sn	Sp	AC	CC	ESn	ESp	(ESn+ESp)/2	ME	WE
P_36_30	0.916	0.996	0.948	0.947	0.082	0.082	0.082	0.058	0.918
G_32_25	0.911	0.988	0.941	0.939	0.084	0.086	0.085	0.085	0.914
est_genome	0.984	0.986	0.985	0.985	0.947	0.961	0.954	0.037	0.026
sim4	1.000	1.000	1.000	1.000	0.986	0.991	0.989	0.007	0.009
spidey	0.885	1.000	0.932	0.930	0.733	1.000	0.867	0.267	0.000
fgene	0.884	0.982	0.920	0.918	0.667	0.950	0.808	0.300	0.050

seq1	Sn	Sp	AC	CC	ESn	ESp	(ESn+ESp)/2	ME	WE
P_36_30	0.636	0.997	0.776	0.743	0.033	0.032	0.033	0.154	0.968
G_32_25	0.765	0.956	0.832	0.823	0.021	0.022	0.022	0.131	0.978
est_genome	0.953	0.991	0.965	0.964	0.705	0.731	0.718	0.063	0.269
sim4	0.773	0.993	0.856	0.838	0.475	0.483	0.479	0.085	0.517
spidey	0.617	0.950	0.744	0.727	0.180	0.233	0.207	0.190	0.767

seq2	Sn	Sp	AC	CC	ESn	ESp	(ESn+ESp)/2	ME	WE
P_36_30	0.509	0.982	0.669	0.613	0.016	0.017	0.017	0.204	0.983
G_32_25	0.432	0.750	0.522	0.493	0.000	0.000	0.000	0.371	1.000
est_genome	0.635	0.813	0.689	0.674	0.316	0.340	0.328	0.335	0.496
sim4	0.355	0.650	0.460	0.423	0.123	0.120	0.121	0.469	0.544

seq3	Sn	Sp	AC	CC	ESn	ESp	(ESn+ESp)/2	ME	WE
P_36_30	0.349	0.983	0.574	0.496	0.011	0.011	0.011	0.324	0.989
G_32_25	0.262	0.691	0.381	0.340	0.000	0.000	0.000	0.496	1.000
est_genome	0.619	0.792	0.669	0.660	0.347	0.395	0.371	0.340	0.405
sim4	0.191	0.593	0.348	0.277	0.033	0.031	0.032	0.672	0.569

Fig. 11. Tabulated Results of DNA only/Protein Version, Sim4, Est2genome, Spidey, Fgenesh-C

vided in figure 11. These results are plotted in figure 12. "P-36-30" clearly has the best specificity followed by Est2genome. Sim4 has high specificity for data sets with closely related organism (data set 1), degrading into the least specific with increasing distance. Fgenesh-C has comparable specificity with the rest of the programs with data set 0. As the data set distance increased, Fgenesh-C was not able to provide any prediction. Similarly, Spidey was only able to provide predictions up to data set 1. For correlation coefficient and sensitivity, Est2genome outperformed all the other programs. Sim4 has the second best performance in cases with close distances. "P-36-30", however, gave the second best sensitivity and correlation coefficient with increasing data set distances. Both Fgenesh-C and Spidey has worse results than "P-36-30", Sim4, and Est2genome. It should be noted that only a small data set was used to generate Spidey and Fgenesh-C data. At the exonic level, Est2genome has the best specificity performance overall. Est2genome also has the best exonic level sensitivity for all but the first data set.

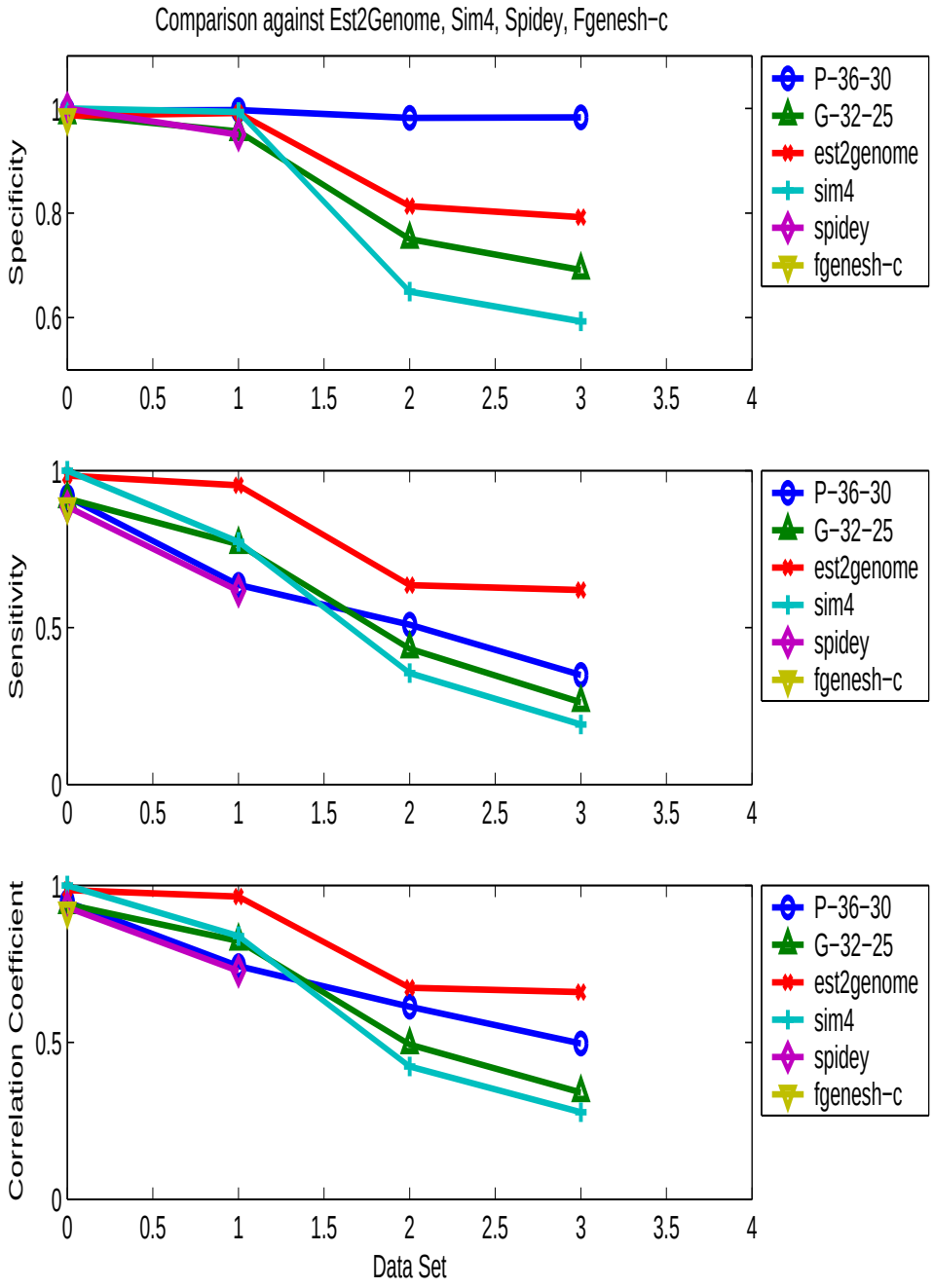

Fig. 12. Comparison against Est2Genome, Sim4, Spidey, Fgenesh-c

CHAPTER IV

SUMMARY AND CONCLUSION

The main purpose of this thesis is to design an algorithm that will return close to 100 percent correct prediction of the exonic regions. The proposed algorithm is a similarity based algorithm which capitalizes on the fact that similar sequences bear similar functions. Similarity based approaches have gained increasing popularity with the recent vast increase in genomic data in GenBank. The proposed algorithm, like most other similarity based algorithms, is based on dynamic programming.

Given a genomic DNA, $X = x_1 \cdots x_n$ and a closely related cDNA, $Y = y_1 \cdots y_n$, these sequences are globally aligned with matching pairs stored in a data set. These indexes of matching sets contain a large jumble of all matching pairs, with a lot of cross over indexes. Dynamic programming is again used to retrieve the longest common non-crossing subsequence from the collection of matching fragments in the data set. In order to improve the statistical performance of the algorithm, donor and acceptor site pattern matching were used to refine the exon-intron boundary. Small fragments, most of which are wrong matches, are dropped. Up to this point, a specificity of very close to 1 was still not achievable. The last, and the most important step, produced the desired specificity of 1 without compromising the correlation coefficient. The initial DNA and cDNA k-mer are translated into amino acid sequences. The translations are also frame shifted with the best match stored in the data set for the second alignment.

This algorithm was implemented in Java on the Unix platform. Statistical comparisons were made against other software programs in the field. Statistical evaluation at both the DNA and exonic level were made against Est2genome, Sim4, Spidey, and Fgenesh-C. My proposed algorithm, by far, has the best performance in the specificity

category. The proposed gene structure prediction algorithm also has on par results in terms of sensitivity and correlation coefficient. The goal of developing an algorithm to predict exonic regions with close to a 100 percent confidence was achieved.

CHAPTER V

FUTURE WORK

The work done for this thesis has laid the groundwork for an approach to predict incomplete exonic regions with a high level of correctness. There are a few additional ways to enhance and improve this program.

The use of scoring matrices incorporates the empirical aspect of commonly substituted amino acid. The use of matrices such as the BLOSUM matrix for amino acid will result in a more realistic alignment due to more judicious selection criteria.

Currently, cDNAs of a closely related organism are used as targets for exonic region prediction. Linking the current program to BLAST searches for targets will add to the practicality of the tool. Use the input genomic DNA as an input to BLAST searches which will return several possible targets. Use these targets to align with the genomic DNA with the algorithmic approach given in this thesis. Concatenate the resulting predicted regions. This method should increase the quantity and quality of the predicted exons.

REFERENCES

[1] J. W. Fickett, "Finding genes by computer: the state of the art," *Trends in Genetics*, vol. 12, pp. 316–320, August 1996.

[2] C. Burge, and S. Karlin, "Prediction of complete gene structures in human genomic DNA," *Journal of Molecular Biology*, vol. 268, pp. 78–94, April 1997.

[3] M. S. Gelfand, "Computer prediction of the exon-intron structure of mammalian pre-mRNAs," *Nucleic Acids Research*, vol. 18, pp. 5865–5869, October 1990.

[4] E. C. Uberbacher, R. J. Mural, "Locating protein-coding regions in human DNA sequences by a multiple sensor — neural network approach," *Proceedings of the National Academy of Sciences USA*, vol. 88, pp. 11261–11265, December 1991.

[5] V. V. Solvyev, A. A. Salamov, and C. B. Lawrence, "Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames," *Nucleic Acids Research*, vol. 22, pp. 5156–5163, December 1994.

[6] S. B. Needleman, C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequences of two proteins," *Journal of Molecular Biology*, vol. 48, pp. 443–453, March 1970.

[7] M. S. Waterman, T. F. Smith, W. A. Beyer, "Some biological sequence matrix," *Advance Mathematics*, vol. 20, pp. 367–387, 1976.

[8] X. Huang, and K. M. Chao, "A generalized global alignment algorithm," *Bioinformatics*, vol. 19, pp. 228–233, September 2003.

[9] R. Mott, "EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA," *Computer Applications in the Biosciences*, vol. 13, pp. 477–478, August 1997.

[10] L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller, "A computer program for aligning a cDNA sequence with a genomic DNA sequence," *Genome Research*, vol. 8, pp. 967–974, September 1998.

[11] S. .F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, May 1990.

[12] S. -H. Sze, and P. A. Pevzner, "Las Vegas algorithms for gene recognition: suboptimal and error-tolerant spliced alignment," *Journal of Computational Biology*, vol. 4, pp. 297–309, Fall 1997.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Cambridge, Massachusetts: The MIT Press, Second Edition, 2001.

[14] M. Burset, and R. Guigo, "Evaluation of gene structure prediction programs," *Genomics*, vol. 34, pp. 353–375, Jun 1996.

[15] M. S. Gelfand, A. A. Mironov, and P. A. Pevzner, "Gene recognition via spliced sequence alignment," *Proceedings of the National Academy of Sciences USA*, vol. 93, pp. 9061–9066, August 1993.

VITA

See Loong, Chin

Work Experiences

- Teaching Assistant, Computer Science, Texas A&M University, College Station, TX, 2002 - 2003
- Propulsion Systems Engineer, United Space Alliance, Houston, TX, 1999 - 2001
- Aerodynamics Engineer, Boeing, Seattle, WA, 1997 - 1999

Education

- Master of Science, Computer Science, Texas A&M University, 2001 - 2003
- Bachelor of Science, Aeronautical & Astronautical Engineering, Purdue University, 1992 - 1996

Contact Addresses

- *Permanent mailing address:* 235 Margo St., San Antonio, TX 78223
- *email:* s0c7469@cs.tamu.edu s0c7469@genesun.tamu.edu schprock@yahoo.com