A USER FRIENDLY ENVIRONMENT

FOR GENE-FINDING PROGRAM EVALUATION (GFPE)

by

WEICHENG ZHANG

(Under the Direction of EILEEN KRAEMER)

ABSTRACT

A graphical, user friendly environment for GFPE (Gene-Finding Program Evaluation), written in Java, is developed to evaluate the prediction accuracy of gene-finding programs GenScan, HMMGene, GeneMark, Pombe and FFG. This tool aims to simplify and/or automate the process of executing the gene-finding programs on sequences of interest and of collecting and analyzing the results. The GUI is designed to be similar to a spreadsheet table. The user can add, cut, copy or paste gene sequence files and annotation files to the table, add, delete, or modify the GFPE program and parameters, select an area on the table to represent the execution of various gene-finding programs on remote servers, automatically collect the prediction results, and draw bar charts to compare evaluation accuracy at the coding level, exon level and protein level. It provides a convenient, user-friendly environment and an efficient file management and execution system, thus saving the user substantial time.

INDEX WORDS: Gene-Finding Program Evaluation, GFPE, GenScan, GeneMark, HMMGene, FFG, Pombe, Graphical User Interface.

A USER FRIENDLY ENVIRONMENT

FOR GENE-FINDING PROGRAM EVALUATION (GFPE)

by

WEICHENG ZHANG

BENG, Beijing Technology and Business University, China, 1988

MENG, Beijing Technology and Business University, China, 1993

MS, The University of Georgia, USA, 2001

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2003

© 2003

Weicheng Zhang

All Rights Reserved

A USER FRIENDLY ENVIRONMENT

FOR GENE-FINDING PROGRAM EVALUATION (GFPE)

by

WEICHENG ZHANG

Approved:

Major Professor: Eileen T. Kraemer

Committee: John A. Miller
Thiab Taha

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2003

To my family for their love and support

ACKNOWLEDGEMENTS

Special thanks to my major professor Dr. Eileen T. Kraemer for her open minded guidance, direction and patience during the entire course of my Masters study. I also thank Drs. John Miller and Thiab Taha for serving on my advisory committee and providing me with guidance.

I am deeply grateful to Dr. William Kisaalita, my major professor in my Ph.D program at Biological & Agricultural Engineering Department, for his kind support of my study of Computer Science. Special thanks also is given to Mr. Jian Wang for his GFPE command line package, which is run by this graphical user interface.

I would also like to thank my friends and colleagues in the Computer Science Department and Biological & Agricultural Engineering Department, Bo Qian, Nan Li and Hongyu Wang, for their companionship and for sharing their programming experience. I would also like to thank all the people whoever provided their kind help which was available whenever I needed it.

Last, I thank my parents and brothers on the other side of the globe for their constant encouragement and support throughout my life; to them I dedicate my further studies.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Computational gene identification plays an important role in genome projects, and numerous programs have been developed to address this problem. Selecting the best gene-finding program or programs for a new organism or category of sequences can be time-consuming and error-prone, as well as problematic for the following reasons: 1) The approaches used in gene identification programs are often tuned to one particular organism; accuracy for one organism or class of organism does not necessarily translate to accurate predictions for other organisms. 2) The performance of the gene-finding programs may depend on the parameter settings used to perform the analysis. 3) Published evaluations of gene identification programs are often not only limited to a particular organism, but may report only a subset of the available metrics. This use of different metrics by the authors of different gene-finding programs complicates the comparison of results. The effort required to reproduce these studies to verify the results or to generate a consistent set of metrics is typically prohibitive.

Numerous gene-finding programs have been developed over the last decade. Despite these limitations, existing methods of gene prediction and models of gene structure are still frequently applied to the newly sequenced organisms, for which no model or method has yet been tuned. Some gene finding programs work well for the gene prediction, even they were not specifically developed for the newly sequenced organisms. To find the best gene-finding program from the numerous existing methods, a lots of trivial and tedious repetitive work need to be done. Thus, it is important to have a rapid and reliable means

to assess the accuracy of different gene identification methods and parameter settings when beginning a new genome project or evaluating a new gene identification program.

Recently, members of the Kraemer Lab at the University of Georgia evaluated several commonly used gene-prediction programs GenScan, HMMGene, GenMark, Pombe and a self developed program FFG, to compare the ability of these programs to accurately predict gene structure for a particular organism, *Neurospora crassa* (Kraemer, 2001). Executing these programs on the test sequences, collating the results of the various programs, and calculating statistics were found to be both time-consuming and error-prone, and motivated the need for development of a standard tool to perform such studies. In 2002, a Gene-finding Program Evaluation (GFPE) program written in Java was developed that aims to produce a command line tool to support the task of evaluating gene-finding programs (Wang et al. 2003). The evaluation criteria employed in this package are based on those described in (Burset et al. 1996). This GFPE program saves the evaluators of gene-finding programs substantial effort in calculating the prediction accuracy.

In 2003, a user friendly environment for GFPE was developed to simplify and/or automate the above process of executing the gene-finding programs on the sequences of interest and of collecting and analyzing the results. The GUI is designed to be similar to a spreadsheet table. The user can add, cut, copy or paste the sequence files and annotation files to the table, add, modify or delete a column in which the selected program and parameters were setup, select an area on the table to represent the execution of various gene-finding programs on remote servers, automatically collect the prediction results from the remote servers and convert them into standard Gene-Finding Features (GFF)

(Sanger Center: GFF, 2000) for further accuracy evaluation. This tool also can draw bar charts to compare the prediction accuracy at the coding level, exon level and protein level.

To run these gene-finding programs in the absence of the tool described in this thesis, users would need go to different websites, paste gene sequence or attach a sequence file, wait for the result to come back, save the result, manually convert its format, and then do the evaluation. Using this tool, the users will not need to experience the above tedious process and, all the work can be done in one place. The benefit of this user friendly environment is that it provides a convenient and efficient file management and execution system, allows users to add and edit gene sequences on the table, setup gene-finding programs and parameters, simultaneously run multiple programs on the selected gene sequences, monitor the running process, automatically collect the prediction results from the remote server, and write the format converted result files to the local hard disk. Also it allows users to save their own work, display files to see the results or visually compare the evaluation results by drawing bar charts.

CHAPTER 2

RELATED WORK

2.1 Gene Terminology

Gene: The functional and physical unit of heredity passed from parent to offspring through mitosis. Genes are pieces of DNA, and most genes contain the information for making a specific protein.

DNA: Deoxyribonucleic acid is a double-stranded helix of nucleotides that carries the genetic information of a cell. It encodes the information for the proteins and is able to self-replicate.

RNA: Ribonucleic acid is an information encoded strand of nucleotides, similar to DNA, but with a slightly different chemical structure. There are three main forms of RNA, each with a slightly different function. mRNA (messenger RNA) is the mediating template between DNA and proteins. The information from a particular gene is transferred from a strand of DNA by the construction of a complementary strand of RNA through a process known as transcription. Next, three nucleotide segments of RNA, called tRNA (transfer RNA), which are attached to specific amino acids, match up with the template strand of mRNA to order the amino acids correctly. These amino acids are then bonded together to form a protein. This process, called translation, occurs in the ribosome, which is composed of proteins and the third kind of RNA, rRNA (ribosomal RNA).

Protein: made of long chains of smaller building blocks called amino acids. Amino acids determine the size, shape, and length of protein molecules. They also give protein molecules the ability to coil and uncoil. They are the chemical building blocks from

which the cells, organs, and tissues such as like muscle are made. Proteins also serve double-duty as hormones, enzymes and antibodies, which help fight off invading germs.

Exon: Any segment of a discontinuous gene, the segments of which are separated by introns.

Intron: Introns are sequences of "junk" DNA found in the middle of gene sequences. These sequences are excised before the mRNA is translated into a protein. The function of introns is not known.

Splicing (Malacinski and Freifelder, 2002): See Figure 2.1(a) and (b). Pre-mRNA splicing occurs in large ribonucleoprotein complexes known as spliceosomes, which are assembled from small nuclear ribonucleoprotein particles (snRNP) that recognize specific sequences on the primary transcript. The spliced (mature) mRNA is transported to the cytoplasm, while the lariat (excised portion) remains in the nucleus and is degraded.

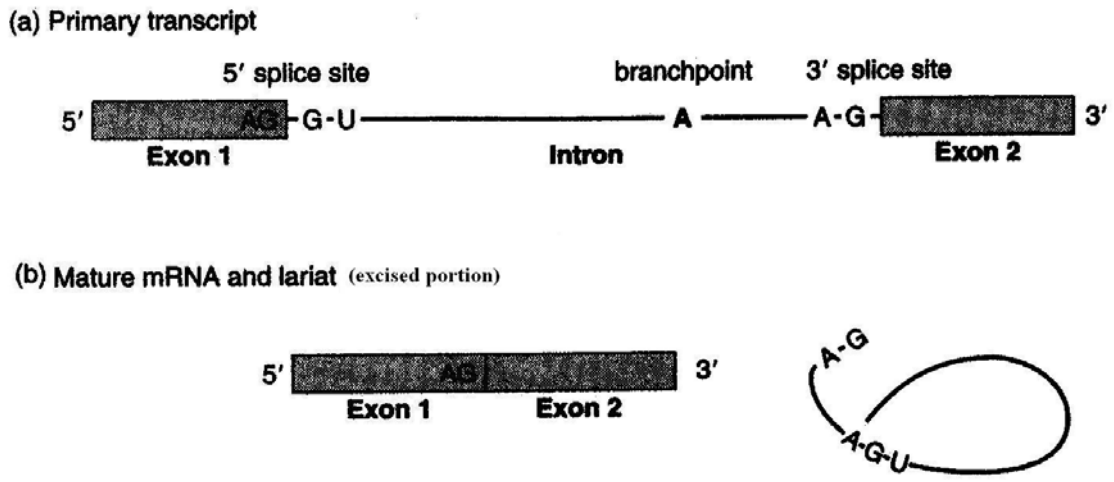

Figure 2.1 The production of mature mRNA by splicing of the primary transcript

2.2 Gene-Finding Programs and Their Approaches

Computational gene identification plays an important role in genome projects. Numerous programs have been developed to address this problem.

In this thesis, a graphical user interface is designed for a program that can evaluate five commonly used gene-finding programs: GenScan (Burge and Karlin, 1997), HMMGene (Krogh, 1997), GeneMark (Borodovsky and McIninch, 1993), Pombe (Chen and Zhang, 1998), and Find Fungal Gene (FFG), developed at the University of Georgia.

GenScan (Burge and Karlin, 1997) is a general-purpose gene identification program that analyzes genomic DNA sequences from a variety of organisms including human, other vertebrates, invertebrates and plants. For each sequence, the program applies a probabilistic model of the gene structure and compositional properties of the genomic DNA for the given organism to determine the most likely gene structure. This model includes consensus sequences involved in transcription and translation, length distributions, and compositional differences. GenScan identifies complete intron/exon structures of a gene in genomic DNA, is able to predict multiple genes, can deal with both partial and complete genes, and can predict consistent sets of genes that occur on either or both strands of DNA. The GenScan program may be accessed through: http://genes.mit.edu/GENSCAN.html. Parameter settings include a choice of organism (vertebrate. arabidopsis, or maize) and a suboptimal exon cutoff value (1.0, 0.50, 0.25, 0.10, 0.05, 0.02, 0.0l).

HMMGene (Krogh, 1997) is a program for prediction of genes in anonymous DNA, designed for prediction of vertebrate and *Caenorhabditis elegans* genes. The program predicts whole genes, and can be used on whole cosmids or even longer sequences. It can

also predict splice sites and start/stop codons. If some features of a sequence are known, such as hits to ESTs, proteins, or repeat elements, these regions can be locked as coding or non-coding and then the program will find the best gene structure under these constraints. The program is based on a hidden Markov model, a probabilistic model of the gene structure. HMMGene can also report the n-best gene predictions for a sequence. This is useful if there are several equally likely gene structures and may even indicate alternative splicing. HMMGene takes an input file with one or more DNA sequences in FASTA format. It also has a few options for changing the default behavior of the program. The output is a prediction of partial or complete genes in the sequences. The output specifies the location of all the predicted genes and their coding regions and scores for whole genes as well as exon scores. The HMMGene program is available at: http://www.cbs.dtu.dk/services/HMMGene/. Through the web page, users may enter sequences, select an organism (vertebrate or *C. elegans*), specify whether or not to predict signals, and specify the number of predictions (1-5) to report.

The **GeneMark** gene prediction software takes several forms. The original GeneMark program (Borodovsky and Mclninch, 1993) relied on inhomogeneous Markov chain models of both coding and non-coding regions, based on analysis of known genes and on the Bayes decision making function, to predict genes in *Escherichia coli* DNA sequences, and was then retrained for *Haemophillus influenzae*, *Mycoplasma genitalium*, and other organisms. GeneMark-Genesis, developed for analysis of organisms such as *Methanococcus jannaschii* and *Helicobaccter pylori*, was designed for the situation in which no experimentally studied segments are available for training. The GeneMark.hmm algorithm (Lukashin and Borodovsky, 1998) generates a maximum-

likelihood parse of the DNA sequence into coding and non-coding regions, and is designed to more precisely locate the exact gene boundaries. This program is available through the web page: http://opal.biology.gatech.edu/GeneMark/eukhmm.cgi. Where the species include a choice of (*H. sapiens, C. elegans, D. melanogaster, A. thaliana, C. reinhardtii, G. gallus, O. sativa, Z. mays, T. aestivum, H. vulgare, M. musculus*), the output options include the choices of (Generate PostScript graphics, Print GeneMark 2.4 predictions in addition to GeneMark.hmm predictions, Translate predicted genes into protein).

The **Pombe** program was developed to find genes and predict exon-intron structure in *Schizosaccharomyces pombe* (Chen and Zhang, 1998). In developing the program, the authors first extracted a training data set from GenBank, checked the annotations for accuracy, and removed redundancy. Execution of the program involves a number of linear discriminant analyses. For example, one analysis differentiates between {sites, introns, exons} and {pseudo sites, pseudo introns, pseudo exons}. Initiation sites, donor sites, and acceptor sites are identified. Exon and intron predictions are the result of the combination of three linear discriminant functions. Other factors considered include oligonucleotide preferences, positional triplet preferences, and the location of ORFs. The results of these intermediate analyses are then combined through dynamic programming to predict gene structure. Pombe is freely available for academic use and is available through the web site at: http://argon.cshl.org/genefinder/Pombe/pombe.htm.

FFG is a pattern-directed program for gene-finding in *N. crassa*, based on statistical analysis of sequence features with genes from *N. crassa* performed by Edelman and Staben (1994). The FFG algorithm begins by identifying possible start and stop sites, as

well as left (5' donor) sites, center (splice branch) sites and right (3' acceptor) sites. Frame numbers are associated with start and stop sites. Any subsequence matching the pattern 'GTRNGT' is identified as a potential left site; any subsequence matching the pattern 'CTRAC' is identified as a potential center site and any subsequence matching the pattern 'YAG' is identified as a potential right site.

Then, the algorithm traverses the list of start sites and builds a list of 'primitive' ORFs. Each ORF ends at the first stop site encountered in the same reading frame in the sequence. At this point, each ORF has one exon. Next, the algorithm repeatedly traverses the ORF list. For each ORF, the algorithm examines the last exon in its list and attempts to extend the ORF to include another exon. This is possible if a splice site can be found within the exon. That is, if the exon contains a 'left' (5' donor) site and both a 'center' (branch site) and 'right' (3' acceptor) site can be found within an acceptable distance (currently set to 300 base pairs). If these are located, another exon is added to the list for that ORF; otherwise, the ORF is marked as complete. Extension terminates when all ORFs are marked as complete.

Finally, the algorithm deletes ORFs that are less than 300 bp in length (an ORF less than 300 bases is not likely to be a gene). When several ORFs overlap, the longest one is selected and the others are deleted. The reverse complement strand is then generated and the process repeated. FFG accepts input sequences in FASTA or plain text format, and produces output in the GFF format (Sanger Center: GFF, 2000), a sequence annotation format developed with gene finding in mind.

2.3 Evaluation Methodology

In general, prediction accuracy can be measured at three levels: at the level of the coding nucleotide, at the level of exonic structure, and the level of the predicted protein product. At the protein product level, the protein encoded by the actual gene is compared with the protein encoded by the predicted gene.

Measurements of accuracy at the coding level compare predicted coding value with the actual coding value for each nucleotide along the test sequence (Figure 2.2). In this widely used approach predictions are divided into four categories:

True Positive (TP) = nucleotides classified as coding in both actual and predicted.

True Negative (TN) = nucleotides classified as non-coding in both actual and predicted.

False Positive (FP) = classified as coding in predicted, but as non-coding in actual.

False Negative (FN) = classified as non-coding in predicted, but as coding in actual.

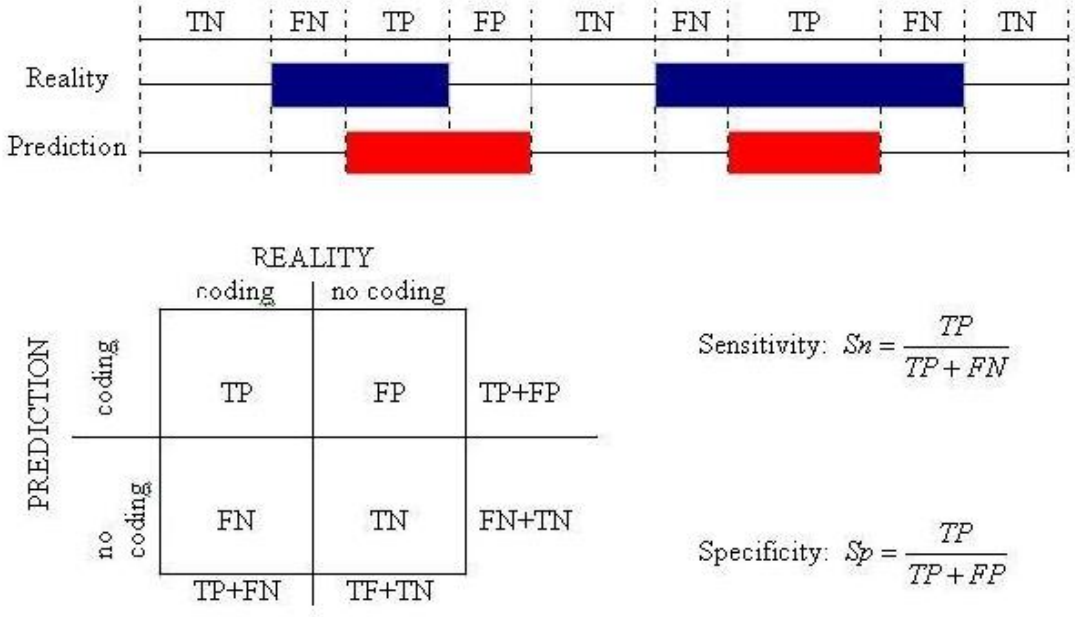

Fig 2.2 Measures of prediction accuracy at nucleotide level (Burset and Guigo, 1996).

Sensitivity is defined as the proportion of coding nucleotides that have been correctly predicted as coding.

Specificity is the proportion of non-coding nucleotides that have been correctly predicted as non-coding.

An issue that arises in evaluating specificity is that the frequency of non-coding nucleotides in genomic DNA sequences is much greater than the frequency of coding nucleotides, so that TN tends to be much larger than FP, with the result of a tendency toward very large non-informative values for specificity. Thus, in much of the literature on gene structure prediction, specificity is instead defined to be TP/(TP+FP), the proportion of predicted coding nucleotides that are actually coding. Other commonly used metrics based on these categories are the Correlation Coefficient (CC), defined as:

$$CC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}$$

the Simple Matching Coefficient (SMC), defined as:

$$SMC = \frac{TP + TN}{TP + FN + FP + TN}$$

the Average Conditional Probability (ACP), defined as:

$$ACP = \frac{1}{4}\left[\frac{TP}{TP + FN} + \frac{TP}{TP + FP} + \frac{TN}{TN + FP} + \frac{TN}{TN + FN}\right]$$

and the Appropriate Correction, AC, defined as:

$$AC = (ACP-0.5)x2$$

(Burset and Guigo, 1996).

While nucleotide-level metrics are often used to evaluate how well the program locates sequence-coding regions, exonic structure metrics are typically used to evaluate

how well the sequence signals (splice sites, start codons, stop codons) are identified (Burset and Guigo, 1996). We evaluate the accuracy of predictions at the exon level by comparing predicted and actual exons along the test sequence (Figure 2.3).

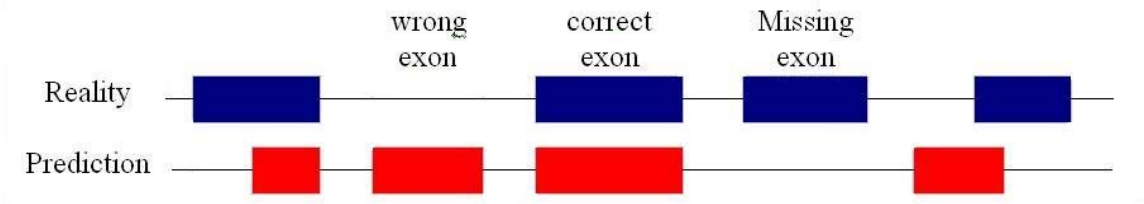

Fig 2.3 Measures of prediction accuracy at the exon level (Burset and Guigo, 1996).

Although this approach is widely used, no unique criterion has been used to consider an exon as 'correctly' predicted. The strictest criterion would score an exon prediction as a correct match only if an exact match exists between actual and predicted start and stop locations, i.e., both splicing boundaries are correctly identified. Kraemer and Wang (2001) label these as 'type 1' predictions. A looser criterion scores a prediction as correct if a partial match occurs, if at least one of the splice sites has been correctly identified. Kraemer and Wang (2001) label these as 'type 2' predictions. Finally, a predicted exon may be scored as correct if the overlap between actual and predicted exceeds some threshold. Kraemer and Wang (2001) label these as 'type 3' predicted exons.

The notions of sensitivity and specificity are still applicable in measurements performed at the exon level. Sensitivity is the proportion of actual exons in the test sequence that are correctly predicted. Specificity is the proportion of predicted exons that are correctly predicted. Also useful are the notions of Missing Exons (ME) and Wrong Exons (WE). ME indicates the proportion of actual exons with no overlap to predicted exons. WE indicates the proportion of predicted exons with no overlap to actual exons.

Kraemer and Wang (2001) developed a method of selecting a threshold for overlap between actual and predicted exons that relies on the notions of overlap-sensitivity and overlap-specificity and an initial empirical evaluation. Overlap-sensitivity is the number of nucleotides in the overlapping region between the predicted exon and the actual exons, divided by the number of nucleotides in the actual exon. Overlap-specificity is the number of nucleotides in the overlapping region, divided by the number of nucleotides in the predicted exon. A Combined Overlap Percentage (COP) was defined to be:

$$COP = \frac{(OverlapSn + OverlapSp)}{2}$$

In reporting the results of their evaluations, Kraemer and Wang (2001) define three categories, labeled one-star (*), two-star (**), and three-star (***). The one-star category includes only the type 1 exons. The two-star category includes only the type 1 and type 2 exons. Both of these categories may be used to evaluate the ability of a program to exactly locate exon and intron boundaries. The three-star category combines this information with a measure of the ability of a program to correctly predict coding regions, and consists of type 1 exons, type 2 exons for which the COP exceeds the threshold (80%), and type 3 exons for which the COP exceeds the same threshold.

2.4 Jian Wang's Work in Evaluation Package

The gene finding evaluation programs used in this GFPE graphical user interface tool to predict gene finding accuracy at the coding, exon and protein levels are a set of Java classes developed by Jian Wang, and using a command line interface. The prediction result GFF format converter programs were also written by Jian Wang. The data used to draw the bar chart in the tool were averaged using Jian Wang's program. The FFG

program was developed by Dr. Eileen Kraemer, Dr. Jinhua Guo and Jian Wang at the Department of Computer Science, the University of Georgia.

CHAPTER 3

A WALKTHROUGH OF THE USAGE OF THE TOOL

Before running the program, this software should be installed, typically on a Windows-based machine. In addition, a dataset consisting of DNA sequences and their annotation files must be prepared. Then the following steps are performed to accomplish the gene-finding program evaluation work.

3.1. Add Files to the "Prediction Files" Table

The system will automatically open a new experiment when it is started. If desired, the user can close this new experiment, and open a previously saved experiment to continue his work.

To add a sequence or sequences to the "Prediction Files" Table, the user must select a single table cell or an area in the first column, click the add file icon in the tool bar, then select single or multiple files, and click "Open". The files will be added to the sequence file column in the order they are selected (Figure 3.1). Similarly, the user may choose the second column to add annotation files. All four tables will have the same sequence and annotation files in the first two columns, regardless of the table through which these files were added.

To see the content of the added files, the user should select them, click the display icon in the tool bar, and the files will be displayed in the bottom text window.

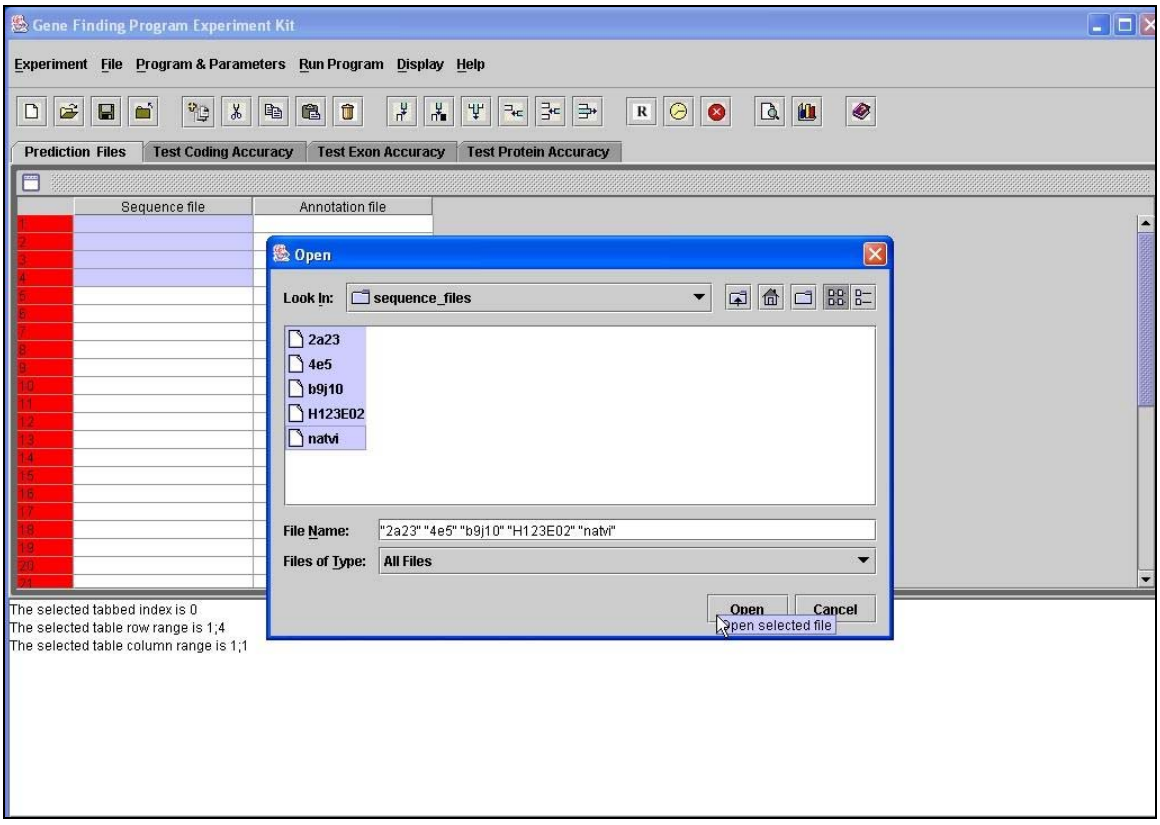

Fig 3.1 Add files to the "Prediction Files" table

3.2 Add Programs and Setup Parameters

To specify the gene-finding programs to be executed, the user may append or insert any of the available gene-finding programs to the table. The user may also set up or modify the program parameters, or delete an unwanted program. The same program may be added multiple times with different parameters.

To add a program, the user should click on the "Program & Parameters" menu, and click "Append Program". A "Select Program and Setup Parameters Window" will popup, as shown in Figure 3.2a. There are five gene-finding programs in this package, they are: GenScan, HMMGene, GeneMark, Pombe and FFG program. The user can select one of them from the drop down JComboBox and setup its parameters, as seen in Figures 3.2b,

3.2c, 3.2d, 3.2e and 3.2f. If the user clicks "Default", the parameters will be set to the default values for that program. If the user clicks "OK", the selected program will be appended to the last column of the table.

To insert a program, the user should select a column, click "Insert Program" in the "Program & Parameters" menu, then select a program and setup its parameters, and click "OK". The selected program will be inserted to the left of the selected column. To modify the program parameters, the user can directly click the column heading in the "Prediction Files" table, and re-setup the parameters from the popup window, or do so using the "Modify Program" choice in the "Program & Parameters" menu. To delete a program, the user should select that program column, and click "Delete Program" in the "Program & Parameters" menu. A popup warning window will ask the user "Are you sure you want to delete the selected column?", the user should click "OK" to delete it. All four tables will have the same programs in the same order. Figure 3.3 shows the table with five gene-finding programs.

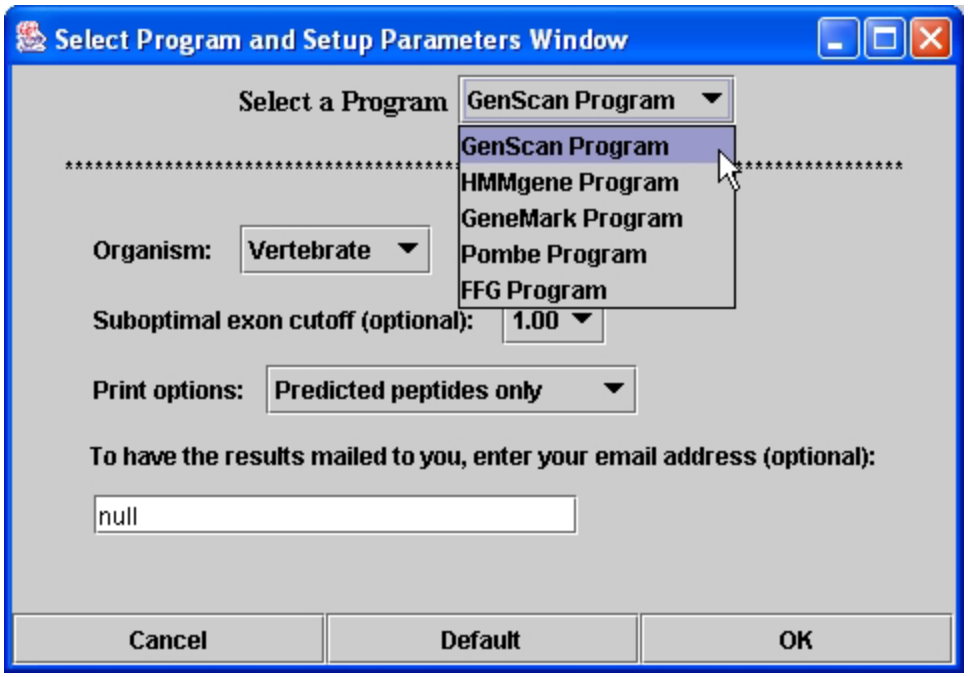

Fig 3.2a Select GenScan program from the drop down JcomboBox

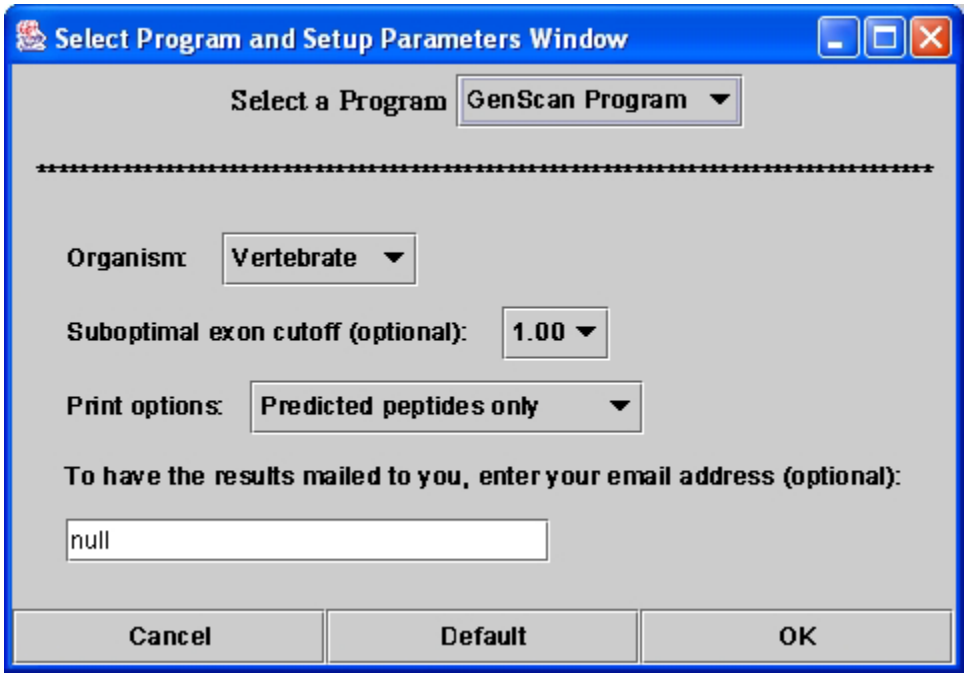

Fig 3.2b Setup Parameters Window for GenScan Program

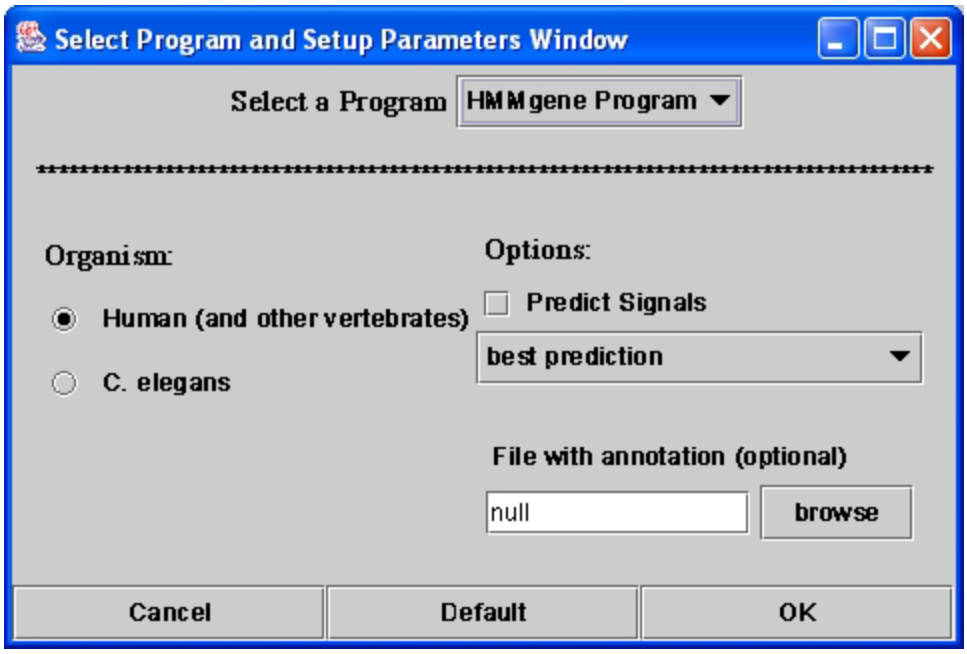

Fig 3.2c Setup Parameters Window for HMMGene Program

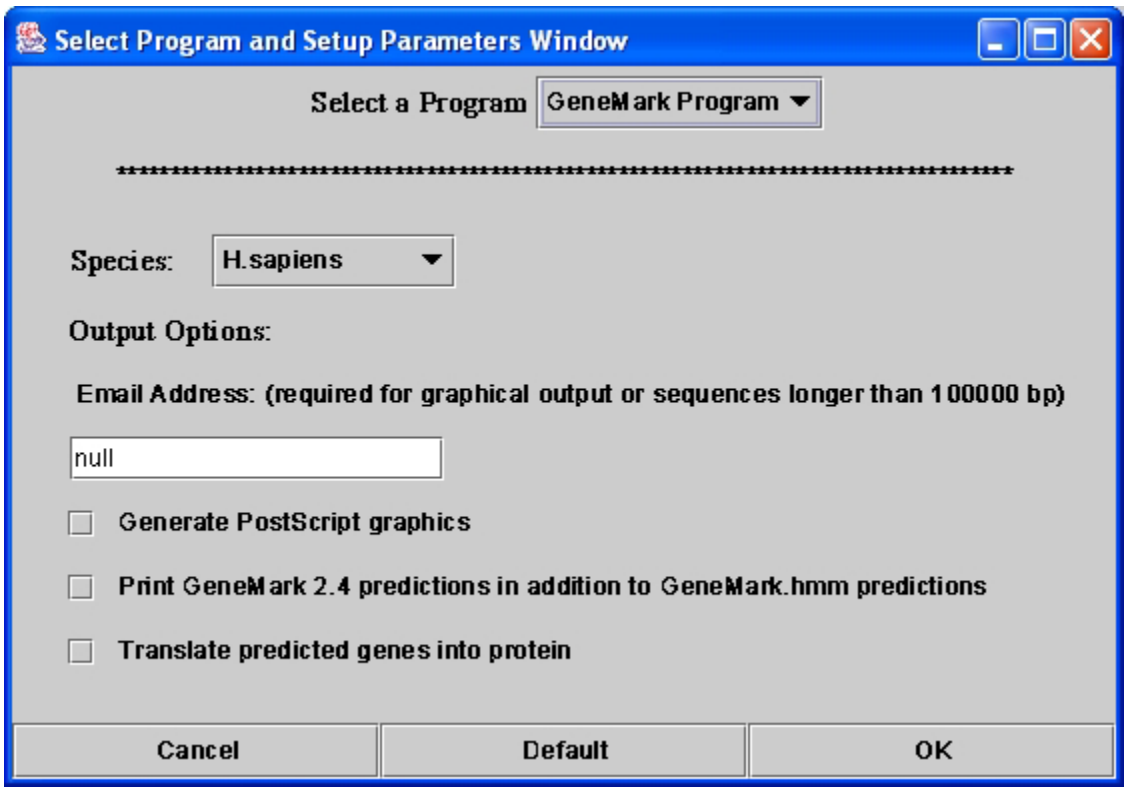

Fig 3.2d Setup Parameters Window for GeneMark Program

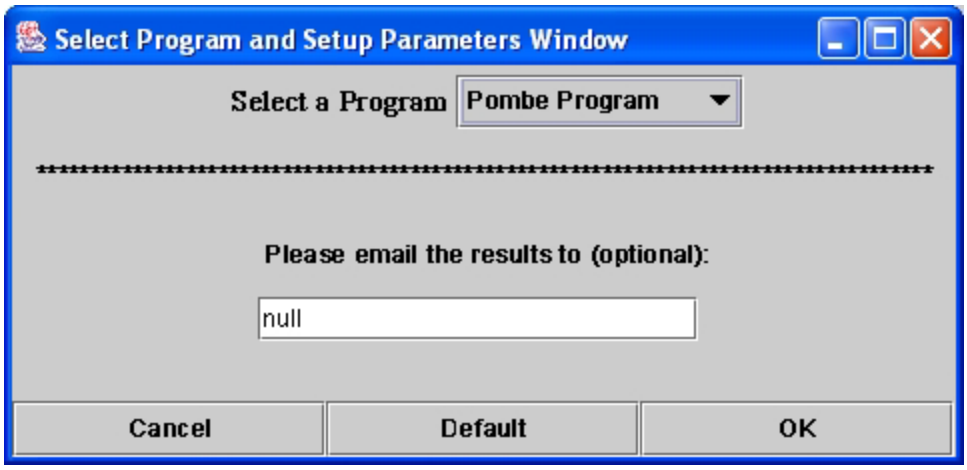

Fig 3.2e Setup Parameters Window for Pombe Program

Fig 3.2f Setup Parameters Window for FFG Program

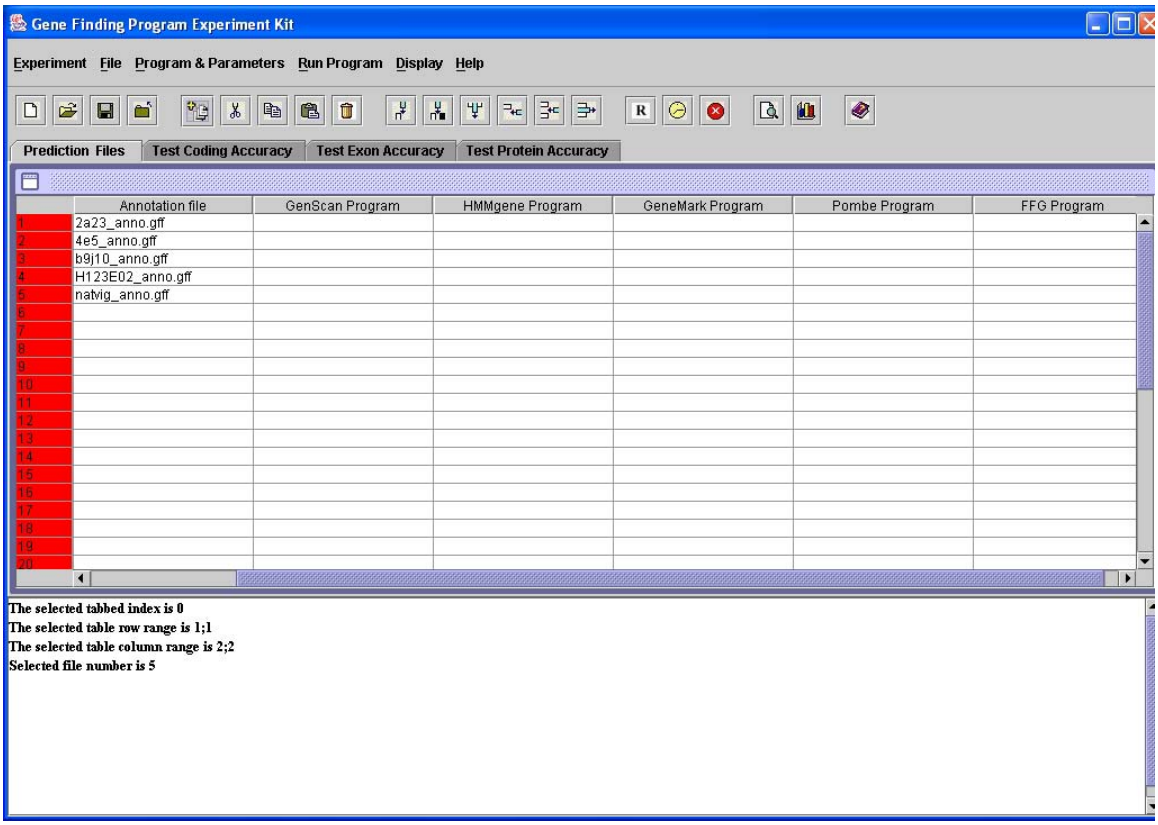

Fig 3.3 A table with five gene-finding programs setup

3.3 Run Programs

Figure 3.4 presents the data flow through the gene-finding evaluation process.

To run the gene-finding programs, the user should select an area in the "Prediction Files" table, click the "Run Program" menu and click on the "Run Prediction". All five gene-finding programs can be run in multiple threads. When the prediction results are available, the system will write the result files to the appropriate prediction_results folder on the hard drive, write the file names to the appropriate table cells and, update the corresponding table cells in the other three tables using the words: "not evaluated yet". The results taken from remote server are converted into GFF format to allow further gene-finding program evaluation.

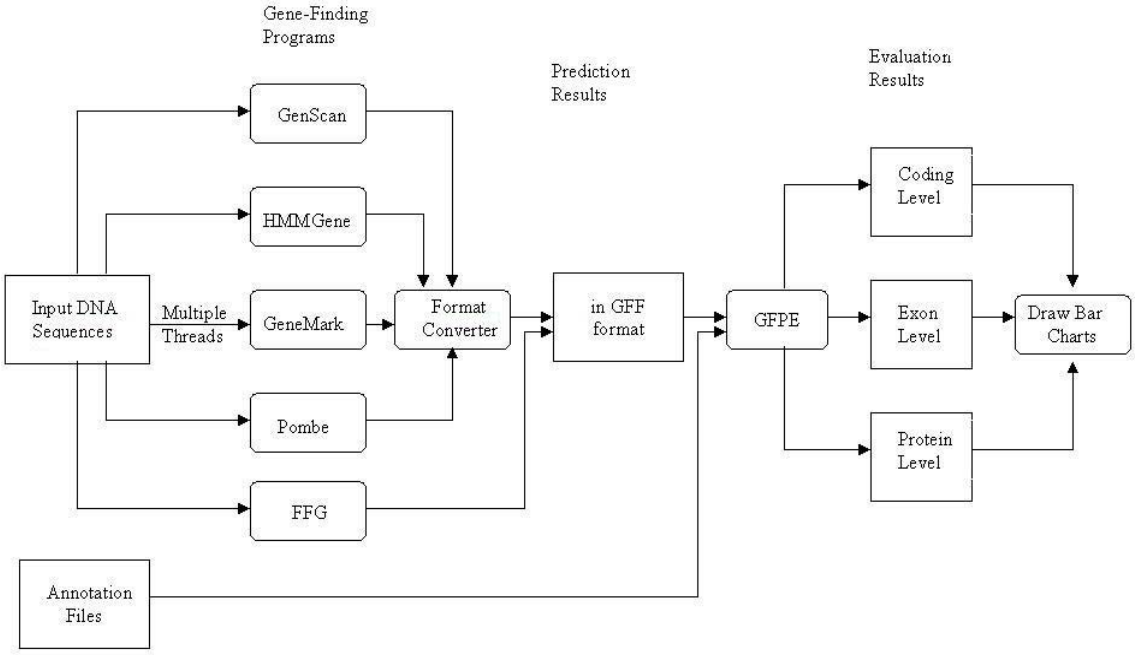

Fig 3.4 Data flow through GFPE

Time required for the execution of the gene-finding programs can vary from a few seconds per file to a few minutes per file, depending on the gene-finding program and the size of the sequence file. Thus, results are not immediately available and the user may wish to monitor the running process.

To monitor the running process, the user should click on the “Run Program” menu and click “Show Running Programs”. The system will update the table to reflect the progress of running programs every 10 seconds in the popup window and display “All Done!” when all the running programs are finished (Figure 3.5).

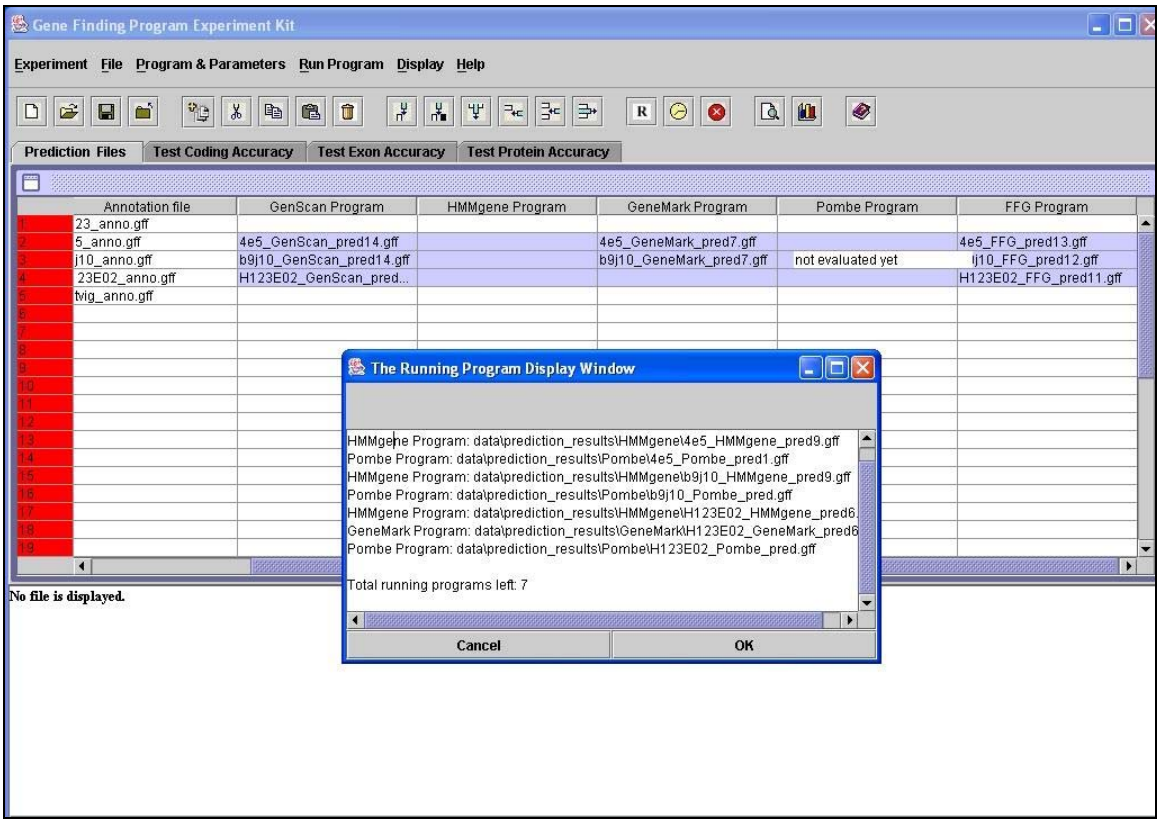

Fig 3.5 Monitor running program window

To evaluate the gene-finding results, the user can choose to test at three levels – the coding level, exon level and protein level, by going to the appropriate table, selecting an area to be tested, clicking on the "Run Program" menu, and then clicking on "Run Evaluation". The system will automatically start the evaluation. As the evaluation results are available, the system will write the result files to the appropriate folder on the hard drive, and update table cells with the corresponding file names. To display the evaluation results, the user can select single or multiple files in an area in the table, click the "Display" menu and click "Display Files". The evaluation results will be displayed in the bottom text window (Figure 3.6).

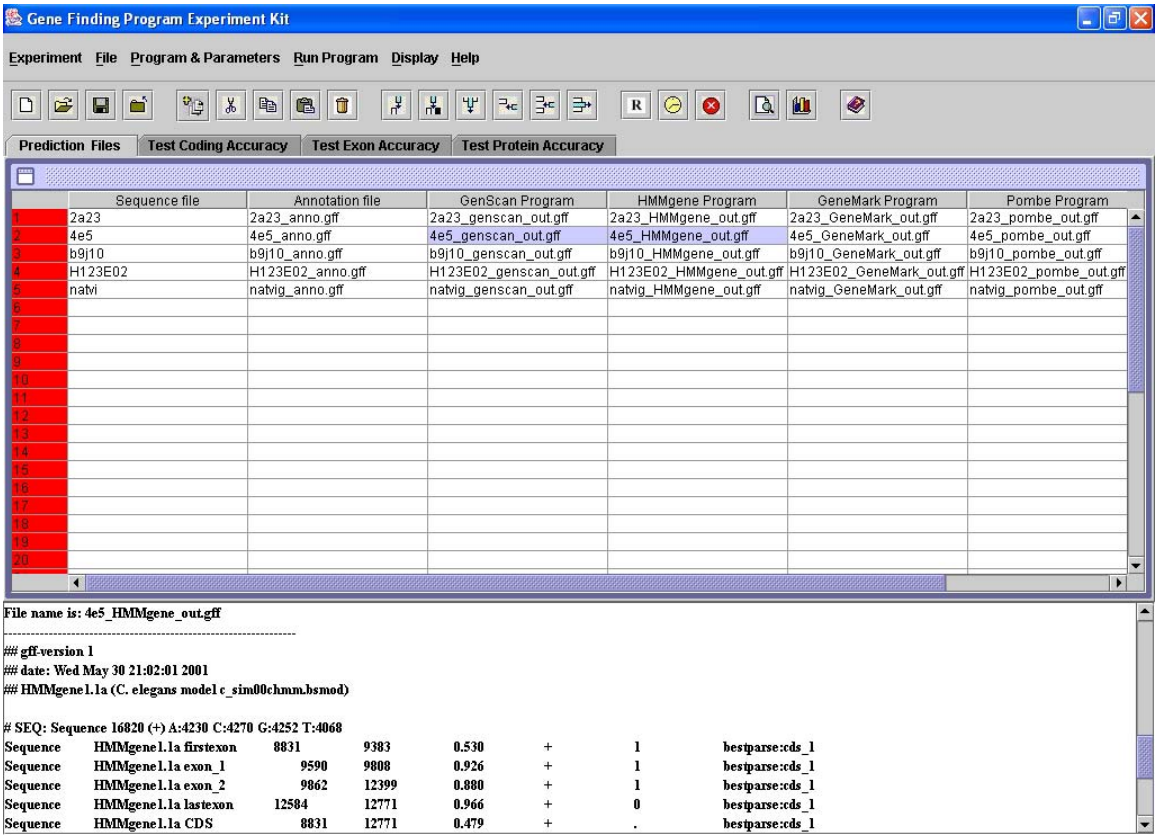

Fig 3.6 Display file content through the bottom text window

3.4 Draw Bar Charts

To compare the evaluation results, the user can view coding, exon or protein level accuracy bar charts. To do so, the user should go to the appropriate evaluation table, select an area that contains the files to be compared, click on the "Display" menu and click on "Draw Bar Chart". A popup window will display the bar chart, as seen in Figure 3.7, 3.8 and 3.9. The user can then click the tabs in the "Coding Level Accuracy Chart" or the "Exon Level Accuracy Chart" to check the different evaluation results.

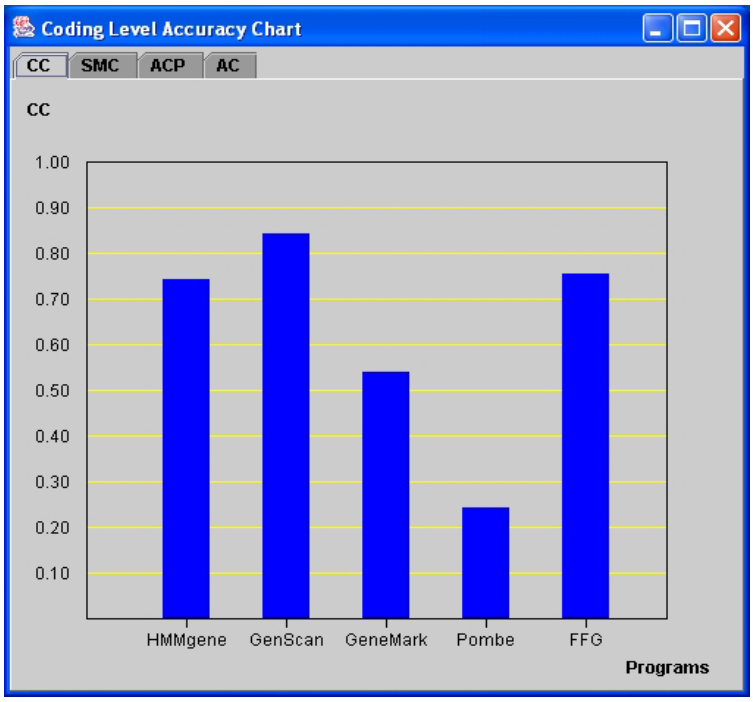

Fig 3.7 Evaluation results bar chart – coding level accuracy

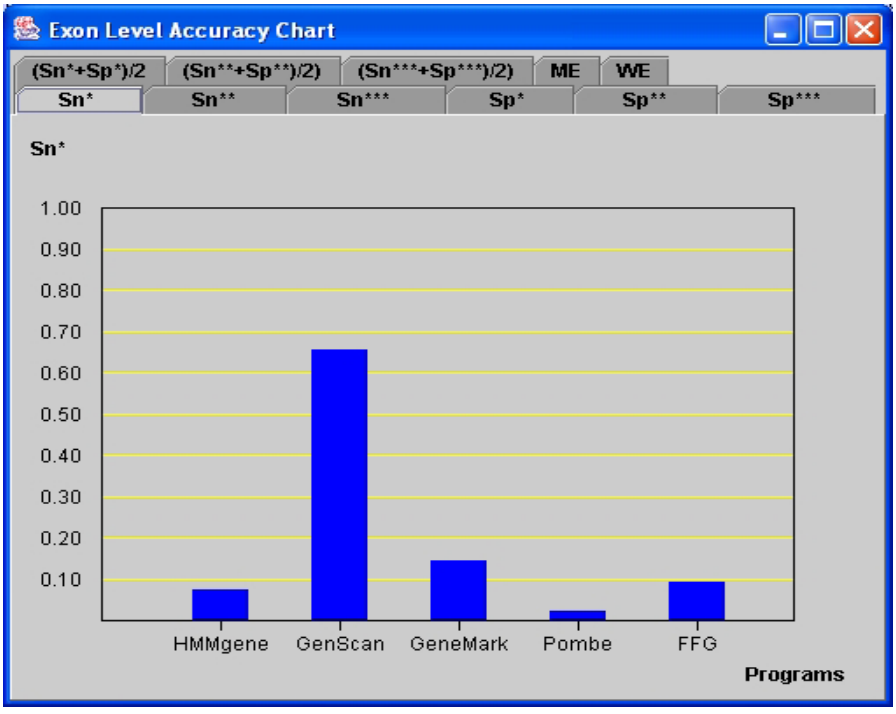

Fig 3.8 Evaluation results bar chart – exon level accuracy

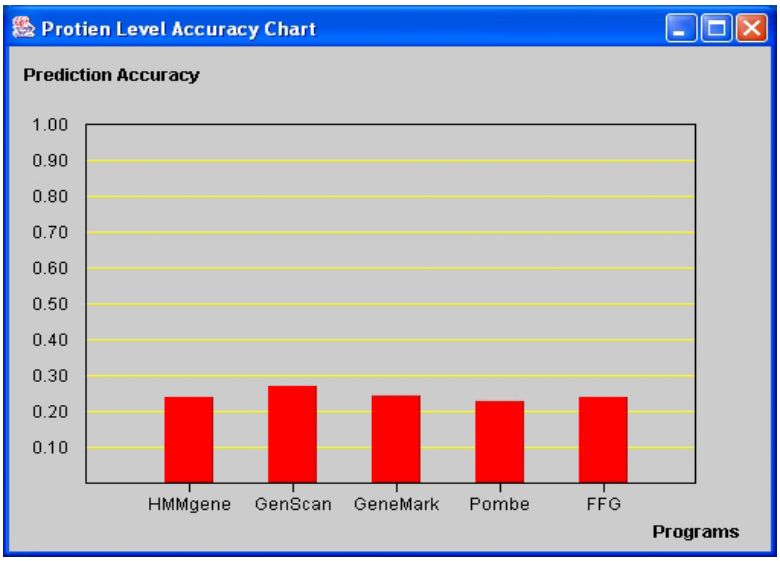

Fig 3.9 Evaluation results bar chart – protein level accuracy

3.5 Summary

This chapter describes the operational features and functionality of GFPE's user friendly environment through a walkthrough usage of the tool, and is intended to serve as a brief user manual. For the detailed user manual, refer to Appendix B.

CHAPTER 4

DESIGN AND IMPLEMENTATION

This software package provides a user friendly environment for GFPE. The main part is the GUI interface that displays the operation menu, tool bar, tables and a text window. To permit users to perform gene-finding program evaluation, the GUI allows users to add, cut, copy, paste or delete files on the table, save or load an experiment, add gene-finding programs and set up the related parameters, run prediction and evaluation functions and compare the evaluation results using bar charts.

Figure 4.1 shows the architecture diagram of this software package. The major operations of "GUI Construction", "Handling of Mouse Events and User Selection", "Data Structures and Operations", "Handling of Sequence Files, Gene-Finding Programs and Parameters", "Gene-Finding Program Prediction", "Gene-Finding Program Evaluation" and "Draw Bar Charts" were implemented using Java, and are described in the following sections of this chapter. To enhance the clarity of discussions of code modules, we use different type faces to distinguish between classes that are part of the Java distribution and those written as an element of this thesis project. The typeface for java.sun.com Java API packages is **bold**, Java classes developed in this project are shown in ***bold italic***, and variables and methods are in plain *italic*.

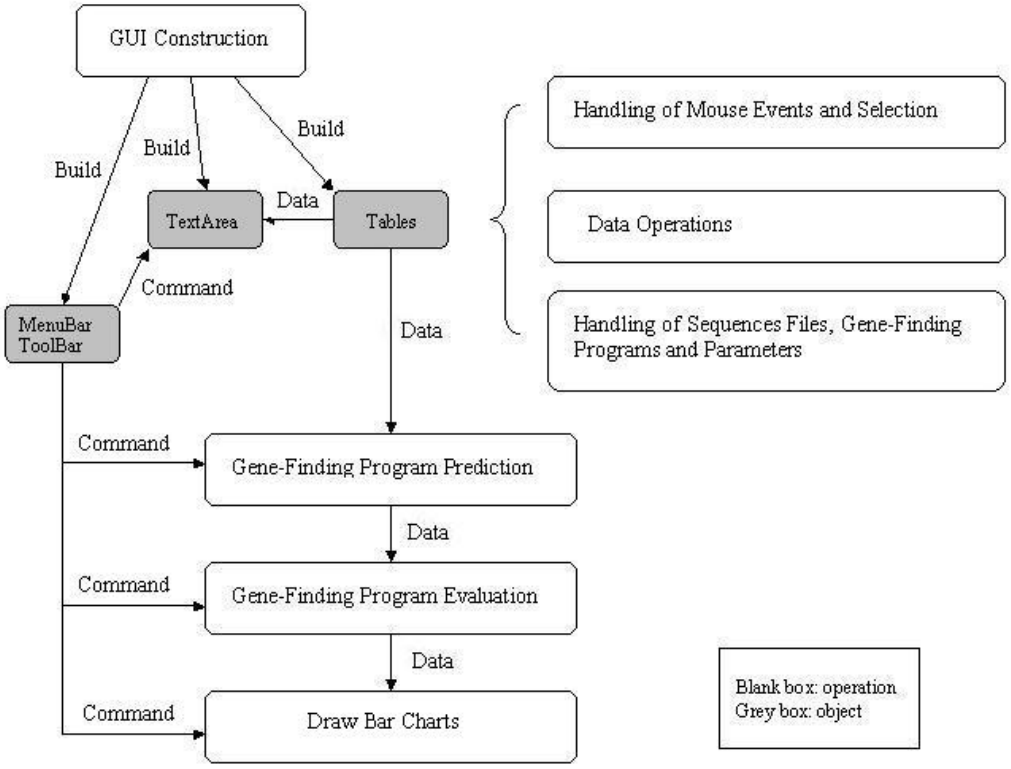

Figure 4.1 The Architecture Diagram of the software package

4.1 GUI Construction

The primary classes involved in the GUI interface are: ***Gui.java, SpreadsheetModel.java*** and ***InternalTableFrame.java.*** The file ***Gui.java*** is the main class of this tool. It creates a **JFrame** and adds a **JMenuBar** to the top and a **JTextArea** to the bottom, and a **JTabbedPane** on which **JInternalFrames** (spreadsheet-like tables) can be created and displayed.

The JMenuBar

The **JMenuBar** holds a **JMenu** of 'Experiment', 'File', 'Table Editor', 'Run Program', 'Display' and 'Help' choices. The **JMenu** contains **JMenuItems,** and may

also contain **JSeparators**. The **JMenu** and **JMenuItems** in this tool are shown in Figure 3.3.

The JToolBar

The **JToolBar** contains five groups of the fast access **JButton**s with image icons.

The JTabbedPane

The **JTabbedPane** holds four ***InternalTableFrame***s that extend **JInternalFrame**. They are the backbone of the tables. Each ***InternalTableFrame*** contains a spreadsheet-like table based on its own ***SpreadsheetModel***. All the tables have the same number of rows and columns, same size table cells and same column headings. The first table is used to display gene sequence file names, annotation file names and gene-finding prediction file names. The other three tables are used to display gene sequence file names, annotation file names and gene-finding program evaluation result file names at the coding level, exon level and protein level. The number of rows in each table initially is set to 40, but the user may append or insert more rows to the table as needed. The number of columns in each table is initially set to 2, but again, the user can append or insert more columns to each table. The maximum number of columns in each table is 100.

The class ***InternalTableFrame*** creates a **JInternalFrame** containing two **JTable**s. Two **TableColumnModel**s are created for the two **JTable**s, one for the row header **JTable** that displays the row numbers in the red background cells, and the other for the spreadsheet-like **JTable** that displays file names in the white background cells. A **JViewport** is created to hold the row header **JTable;** without the **JViewport** the scrolling for the two **JTable**s would not match.

The class ***InternalTableFrame*** also has some accessor methods for the class ***SpreadsheetModel***. It also includes three event listeners: table cell selection listener (*SelectionListener*), column head click listener (*ColumnHeaderListener*) and right click mouse button listener (*MyMouseListener*).

The JTextArea

The **JTextArea** is a multi-line area that is used to display file content or program parameters in this tool. **JScrollPane** is used to make the **JTextArea** scrollable. The font and font size are set to "Serif" and “Font.BOLD, 12”, respectively.

4.2 Handling of Mouse Events and User Selection

The primary class involved in the handling of mouse events and user selections is ***InternalTableFrame.java***. The class ***InternalTableFrame*** includes three private event listener classes: table cell ***SelectionListener***, table cell ***MyMouseListener***, and table head ***ColumnHeadListener***.

The table cell ***SelectionListener*** class implements **ListSelectionListener** and handles table cell selection. Whenever a **ListSelectionEvent** happens, this class will assign the new position values of the selected column starting index, selected column end index, selected row starting index and selected row end index to the variables *colIndexStart, colIndexEnd, rowIndexStart* and *rowIndexEnd*, respectively. These values can then be used to perform appropriate operations such as to add or delete files, display a file, cut, copy or paste files, run programs and draw bar charts.

The table cell ***MyMouseListener*** class extends **MouseAdapter** and handles the **MouseEvent**s that happen on any table cells. Only the right click button function is

implemented in this class. Whenever the right mouse button is clicked on a table cell or a selected area in the table, a ***RightClickMenu*** object will be created, and a small menu is popped up allowing the user to quickly access the file edit buttons: "Add", "Display", "Cut", "Copy", "Paste" and "Delete".

The table head ***ColumnHeadListener*** class extends **MouseAdapter** and handles **MouseEvent**s on the user-appended or user-inserted table column heads (**JTableHeader**s). Whenever a left mouse click or right mouse click event happens on those table column headers, a ***ColumnModify*** object will be created, and a "Program and parameters Modifying Window" is popped up allowing the user to modify the program parameters or change the original program to a new program and set up the related parameters.

4.3 Data Operations

The primary classes involved in management of data structures are ***TableManager.java*** and ***SpreadsheetModel.java***. The class ***SpreadsheetModel*** extends **DefaultTableModel** and implements **Serializable.** It is the backbone of the table data and contains 4 important Vector variables: *columnNameVector, filePathVector, fileNameVector* and *parameterVector*. The ***TableManager*** class allows the user to operate on data, add files to the table, save the tables (actually, save the data Vectors of each table), load a saved experiment and display the original stored file names on the table, and read the user selected files and display them on the GUI text window.

Data Storage

The *columnNameVector* stores column headers - the user-appended or user-inserted program names. Initially it has only three columns: "" (name is null), “Sequence file" and "Annotation file".

The files in each table are stored in appropriate directories on the hard drive, as shown in Figure 3.1. Their paths and names are stored in two different Vectors of a ***SpreadsheetModel.java*** to serve different purposes. One Vector is the *filePathVector* that is used to hold the row data Vector of file paths, in which the file paths are stored in the order of the column index, and the other vector is the *fileNameVector* that is used to hold the row data vector of file names, in which the file names are stored in the order of the column index. The files in *filePathVector* are used when its associated table is selected and the gene-finding programs or evaluation programs are called. The files in *fileNameVector* are used only for displaying the file names in that associated table.

The *parameterVector* stores the parameters of the user-selected program. The parameters are stored in the **String** format based on the column index; they are null at the index position 0, 1 and 2, since those places correspond to the row header, “Sequence file” and “Annotation file”, which have no programs. The maximum number of parameters that the *parameterVector* can store in each table is set to 100.

Data operation

All the data operations are through the ***SpreadsheetModel***. The ***TableManager*** class is a major class controlling data operations such as adding files to the table, saving tables, opening a saved table file or displaying the selected prediction or evaluation files. When saving the tables, the four Vectors of *columnNameVector, fileNameVector,*

filePathVector and *parameterVector* in the ***SpreadsheetModel*** are written into a user named file in Objects by **ObjectOutputStream**. When loading an experiment, the stored table data are restored by **ObjectInputStream**, four new ***SpreadsheetModel***s are created using the restored data, and four tables are constructed using a method of *createInternalTabs(model)* in ***Gui.java*** with the newly obtained ***SpreadsheetModel***s.

The ***ColumnAppend, ColumnDelete, ColumnInsert, ColumnModify, FileEdit, Go, GoColdSpringHarbor, GoDenmark, GoGaTech, GoMIT, RowsAppend*** and ***RowsEdit*** classes will update some or all Vectors of *columnNameVector, fileNameVector, filePathVector* and *parameterVector* if these Objects are called and executed.

4.4 Handling of Sequence Files, Gene-Finding Programs and Parameters

The primary classes involved in handling of sequence files, gene-finding programs and parameters set up are ***FileEdit.java, ReadFile.java, WriteToFile.java, ColumnAppend.java, ColumnDelete.java, ColumnInsert.java*** and ***ColumnModify.java.*** The class **FileEdit** allows the user to add sequence files and annotation files to the table by selecting an area in the first two columns of any of the four tables, and also to cut, copy or paste files in the first two columns, or to delete files in all the columns. The file data can be accessed by the class ***ReadFile***, and can be saved by the class ***WriteToFile***.

The class ***ColumnAppend*** is a subclass of Java's **JFrame** that contains five gene-finding programs and related parameter **JPanel** windows for appending a program to the four tables. The class ***ColumnDelete*** can delete a selected program column. The class ***ColumnInsert*** can insert a program to the left of the selected column in the four tables.

The class ***ColumnModify*** allows the user either to change a selected program or modify the parameters without changing the program.

4.5 Gene-Finding Program Prediction

The primary classes involved in running the prediction functions are ***Go.java, GoColdSpringHarbor.java, GoDenmark.java, GoGaTech.java, GoMIT.java, GoFFG.java, HttpUpload.java, ThreadsRuning.java*** and ***ThreadsStop.java.*** The ***Go*** class is designed to run the appropriate prediction or evaluation program, based on the user-selected table. If the user selects the first table – "Prediction Files" table, the ***Go*** class will run the appropriate gene-finding programs by checking the table column headers of the selected area, if the column header is "GenScan Program" ***GoMIT*** will be run using the selected gene sequences.

Whenever a new prediction file is to be written to the hard drive, the ***Go*** class will examine whether the new prediction file's name exists through the *checkFileExist_RenameItIfExist (File myFile)* method. If it exists, the new prediction file will be renamed to avoid overwriting.

There are five gene-finding programs. GenScan, HMMGene, GeneMark and Pombe are on remote web servers, to which our program must upload the gene sequence files and the user-selected parameters. The upload methods include either setting up a **URLConnection** to the remote servers, and using the "name=value" pairs *POST* method to transfer the sequence file and the parameters to the remote servers, or using the ***HttpUpload*** class to set up an **HttpURLConnection** to the remote server, then packing up the file and the paramters into a multi-part upload bundle, and posting the sequence

file and the parameters to the remote servers. The transferred data are queued in the servers to be calculated. When the prediction result is available, **InputStreamReader** and *getInputStream()* are used to get the result back.

GenScan, HMMGene, GeneMark, Pombe and FFG programs can be run by the ***GoMIT, GoDenmark, GoGaTech, GoColdSpringHarbor*** and ***GoFFG*** class, respectively. These classes are implemented using multithreading. The class ***ThreadsRunning*** can monitor the running threads by checking if the threads are alive or not, and updating the information every 10 seconds on a popup window until all the programs have finished. The class ***ThreadsStop*** can terminate the running programs. A *stopFlag* is set to true to stop the active multithreaded execution in the case that the user does not want to continue running the prediction function. The *stopFlag* is placed near the end of the prediction execution function in the ***GoColdSpringHarbor, GoDenmark, GoMIT, GoFFG, GoGaTech*** classes, just before writing the prediction result into a file.

The GenScan program, on the Burge Laboratory web server at Massachusetts Institute of Technology, is run by the ***GoMIT*** class using **HttpUpload.** The parameters and file are posted through multipart/form-data.

The HMMGene program, on a server at the Center for Biological Sequence Analysis, Technical University of Denmark, is run by the ***GoDenmark*** class using **HttpUpload.** Again, the parameters and file are posted through multipart/form-data.

The GeneMark program, on a server at the School of Biology, Georgia Institute of Technology, is run by the ***GoGaTech*** class using **URLConnection.** The parameters and file are posted through “name=value” pairs.

The Pombe program, on a server at the Zhang Lab, Cold Spring Harbor Laboratory, is run by the ***GoColdSpringHarbor*** class using **URLConnection.** The parameters and file are posted through "name=value" pairs.

The FFG program, written in C++ code and compiled into an executable file FFG.exe, is run on the local drive by the Java class ***GoFFG,*** implemented in multiple threads using the Java *Runtime.exec(File file)* method.

4.6 Gene-Finding Program Evaluation

The primary class involved in running the evaluation function is ***Go.java.*** A set of other major classes in the GFPE directory are provided by Jian Wang, and have been compiled into Java class files. The ***Go*** class is designed to run appropriate prediction or evaluation programs based on the user-selected table. To run evaluation programs, one needs to set up the appropriate class path in the system environment variables. If the user selects the second table - "Test Coding Accuracy" table, ***Go*** will execute *TestCodingAccuracy.execute(args),* where *args* is a string that contains the names of the gene sequence file, annotation file and the prediction result file from the first table. If the user selects the third table - "Test Exon Accuracy" table, ***Go*** will execute *TestExonAccuracy.execute(args),* where *args* is a string that contains the names of the gene annotation file and the prediction result file from the first table. If the user selects the fourth table - "Test Protein Accuracy" table, ***Go*** will execute *TestProteinAccuracy.execute(args),* where *args* is a string that contains the names of the gene sequence file, annotation file and the prediction result file from the first table.

Whenever a new evaluation file is to be written to the hard drive, the ***Go*** class will examine whether the new evaluation file's name exists through the *checkFileExist_RenameItIfExist (File myFile)* method. If it exists, the new evaluation file will be renamed to avoid overwriting.

4.7 Draw Bar Charts

The primary classes involved in drawing bar charts are ***DrawBarChartDataProcess.java*** and ***DrawBarChart.java.*** The ***DrawBarChartDataProcess*** class processes data files based on the user-selected table index and the user-selected area in the table, and calculates the weighted average if multiple rows were selected. The weighted average is described in detail in the ***DrawBarChartDataProcess.java*** file. A **Vector** array is used to store the averaged data. For example, *yAxisData_CodingVector* is used to hold "CC", "SMC", "ACP" and "AC" average data in the coding bar chart, *yAxisData_exonVector* is used to hold "Sn*", "Sn**", "Sn***", "Sp*", "Sp**", "Sp***", "(Sn*+Sp*)/2", "(n**+Sp**)/2)", "(Sn***+Sp***)/2)", "ME" and "WE" average data in the exon bar chart.

The ***DrawBarChart*** class uses the **Graphics** class methods *drawLine, drawstring* and *fill3DRect* to draw the X axis, Y axis and bars. The number of bars to be drawn in the chart depends on the selected columns (programs). The ***DrawBarChart*** class can dynamically adjust the size of the window based on the bar numbers, and make them evenly distributed along the X axis.

4.8 Summary of Classes Implemented

This tool has been developed using object-oriented design. Some of the major classes created are:

1. Command line **GFPE package** (provided by Jian Wang).
2. ***ColumnAppend***: A subclass of Java's **JFrame** that contains 5 gene-finding programs and related parameters for appending a column to the 4 tables.
3. ***ColumnDelete***: Delete a selected column. A warning window will popup to make sure the user delete or not.
4. ***ColumnInsert:*** A subclass of Java's **JFrame** that contains 5 gene-finding programs and related parameters for inserting a column to the left of the selected column in the 4 tables.
5. ***ColumnModify***: A subclass of Java's **JFrame** that contains 5 gene-finding programs and related parameters for modifying a column, the user can either change the program or re-setup the parameters without changing the program.
6. ***DrawBarChart***: A subclass of Java's **JPanel** using **Graphics** class methods *drawLine, drawstring* and *fill3Drect* to draw the X axis, Y axis and bars. It can dynamically adjust the size of the window based on the number of bars, and make them evenly distributed along the X axis.
7. ***DrawBarChartDataProcess***: Processes data files based on the selected table and the selected area in the table, calculates the weighted average if multiple rows were selected.
8. ***FileEdit***: Lets the user edit only the "sequence file" column and "annotation file" column by selecting an area in the two columns at any one of the 4 tables, to

perform file Cut, Copy or Paste in the two columns. Delete can be performed in all the columns.

9. ***Go***: Runs the appropriate prediction or evaluation programs based on the user selected table. If the user selects the second table - "Test Coding Accuracy" table, ***Go*** will execute *TestCodingAccuracy.execute(args),* where *args* is a string that contains the names of the gene sequence file, annotation file and the prediction result file from the first table. If the user selects the third table - "Test Exon Accuracy" table, ***Go*** will execute *TestExonAccuracy.execute(args),* where *args* is a string that contains the names of the gene annotation file and the prediction result file from the first table. If the user selects the fourth table - "Test Protein Accuracy" table, ***Go*** will execute *TestProteinAccuracy.execute(args),* where *args* is a string that contains the names of the gene sequence file, annotation file and the prediction result file from the first table.
10. ***GoColdSpringHarbor***: Runs the Pombe program using multithreading on a remote server at the Zhang Lab, Cold Spring Harbor Laboratory.
11. ***GoDenmark***: Runs the HMMGene program using multithreading on a remote server at the Center for Biological Sequence Analysis, Technical University of Denmark.
12. ***GoFFG***: Runs the FFG program on local drive using multithreading. The FFG program is a set of C++ codes written by Kraemer et al., they were compiled into an executable file FFG.exe for use in this package.
13. ***GoGaTech***: Runs the GeneMark program using multithreading on a remote server at the School of Biology, Georgia Institute of Technology.

14. ***GoMIT***: Runs the GenScan program using multithreading on the Burge Laboratory web server at Massachusetts Institute of Technology
15. ***Gui***: The start-up class of this tool. It creates a **JFrame** and adds a **JMenuBar** to the top and a **JTextArea** to the bottom, and the rest of the **JFrame** is used for a **JTabbedPane** on which **JInternalFrames** (spreadsheet-like table) can be created and displayed.
16. ***HttpUpload***: A helper class to package up files and parameters as a multi-part upload bundle to a web-server.
17. ***InternalTableFrame***: Creates **JInternalFrames** containing two **JTable**s. Two **TableColumnModel**s are created for the two **JTable**s. One for the row headers and the other for the spreadsheet-like table. A **JViewport** is created to hold the row header table. It has some accessor methods for the class ***SpreadsheetModel.*** It also includes three event listeners: table cell selection listener, column head click listener and right click mouse button listener.
18. ***ReadFil***e: Reads in a file from the designated directory on the hard disk.
19. ***WriteToFile***: Writes a new file to the designated directory on the hard disk.
20. ***RightClickMenu***: A small menu allowing the user to quickly access the file edit buttons: "Add", "Display", "Cut", "Copy", "Paste" and "Delete".
21. ***RowsAppend***: Appends rows to the end of the 4 tables.
22. ***RowsEdit***: A subclass of Java's **JFrame** that lets users specify how many rows to be inserted into the current table, or delete the selected rows from the table. For both insertion and deletion, the operation is applied to all four tables.

23. ***SpreadsheetModel***: The backbone of the table data, it contains four important Vector variables: *columnNameVector, filePathVector, fileNameVector* and *parameterVector,* which are used to store column name, file path and name, file name and parameters.
24. ***TableManager***: Allows the user to operate on data, add files to the table, save the tables (Actually, save the data Vectors of each table), open a saved file to display the original stored file names on the table, read the user selected files and display them on the GUI text window.
25. ***ThreadsRunning***: Checks on running threads and updates information about the running threads on a popup window every 10 seconds.
26. ***ThreadsStop***: A *stopFlag* is set to true to stop the active multithreaded execution in the case that the user does not want to continue running the prediction function. The *stopFlag* is placed near the end of the prediction execution function in the ***GoColdSpringHarbor, GoDenmark, GoMIT, GoFFG, GoGaTech*** classes, before writing the prediction result into a file.

CHAPTER 5

USER EVALUATION

For GUI-based software designed for use by scientific researchers, the life-cycle development relies heavily on user requirements, user evaluation and user feedback. Accordingly, during the development of this tool, user evaluation was obtained. The users were researchers and graduate students at the University of Georgia. Users were asked to use the tool and feedback was obtained.

GFPE User Friendly Environment, version 1.0

Version 1.0 of the GUI dealt with the appearance of the menu bar, four spreadsheet-like tables attached by the tabs of "Prediction Files", "Test Coding Accuracy", "Test Exon Accuracy" and "Test Protein Accuracy" and a text display window. Interaction with the spreadsheet allows users to select an area in the table, add files to that area, edit the files using cut, copy, paste or delete, run the gene-finding programs and evaluate the prediction results by coding accuracy, exon accuracy, and protein accuracy, and display prediction results or evaluation results in the bottom text window. Each table has only seven fixed columns whose column header showing "Sequence Files", "Annotation Files", "GenScan Program", "HMMGene Program", "GeneMark Program", "Pombe Program" and "FFG Program". After an initial demonstration, user feedback was sought. The interface look was fine and the participants felt that the tool would save users much effort and time, compared to independently running those gene-finding programs and collecting the prediction results. More concrete ideas about the additional features included:

- Enabling dynamic addition, insertion or deletion of programs on the table, allowing users to compare the programs that he/she wants, or to compare the same program with different set of parameters on one table, instead of the initial fixed seven columns.
- Enabling a mouse right click popup menu for user to quickly access the “File” edit menu such as “Add”, “Cut”, “Copy”, “Paste” and “Delete”.
- Enabling a mouse click popup window on the column header to change the program or program parameters and merging the separate parameter window into one window.
- Implementing multithreading to improve the speed of running programs on remote servers and to enable monitoring the running process.
- Drawing bar charts to compare the evaluation results visually.

GFPE User Friendly Environment, Version 2.0

In version 2.0, the additional features described in the previous user evaluation were implemented. The focus was on dynamically appending or inserting the columns to the table, using multithreading to improve the gene-finding program prediction speed and drawing the bar charts. After a demonstration of version 2.0, the user suggested the following additions:

- Enabling undo and redo functions to allow the user to move back and forth between the current and previous status.
- Enabling a print feature to allow the user to print some results.
- Reporting failure if unable to setup a connection, or unable to get the result back when accessing remote servers to run the gene-finding program predictions.

- Enabling one bar chart to compare the prediction evaluation results of the different sequences predicted by the same program.

The above suggestions will be considered in the future work.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In retrospect, the development of the present version 2.0 of GFPE user friendly environment was a good, hands-on, learning experience of programming in the Java language using OOD/OOP. Ease of user interaction was kept in mind throughout the development process. Using a visualization-based software tool for solving problems associated with gene-finding research was found to be quite effective.

- A graphical, user friendly environment for Gene-Finding Program Evaluation (GFPE) written in Java is developed to evaluate the prediction accuracy of gene-finding programs of GenScan, HMMGene, GeneMark, Pombe and FFG program.
- The middle part of the GUI is designed similar to a spreadsheet table. The user can add, cut, copy or paste gene sequence files and annotation files to the table, add, modify or delete a column in which the selected program and parameters were set up, select an area on the table to represent the execution of various gene-finding programs on remote servers, automatically collect the prediction results, and draw bar charts to compare evaluation accuracy at the coding level, exon level and protein level. All the user's prediction and evaluation work can be saved as a file for future use.
- This tool aims to simplify and/or automate the process of executing the runs of the gene-finding programs on the sequences of interest and of collecting and analyzing the results, saves users substantial time and command line typing work.

- It also serves as a basic platform for building further enhancements, which could be used to solve other Bioinformatics applications.

6.2 Future Work

As suggested in the version 2.0 user feedback, we may do the following future work:

- Enabling undo and redo function to allow user back and forth between the current and previous status.
- Enabling print feature allow user to print some results.
- Reporting failure if unable to setup connection, or unable to get the result back when access remote servers to run the gene-finding program predictions.

APPENDIX A

Gene-Finding Features (GFF): Definition (Sanger Center, 2000)

Fields are: <seqname> <source> <feature> <start> <end> <score> <strand> <frame> [group]

<seqname>

The name of the sequence. Having an explicit sequence name allows a feature file to be prepared for a data set of multiple sequences. Normally the seqname will be the identifier of the sequence in an accompanying fasta format file. An alternative is that 'seqname' is the identifier for a sequence in a public database, such as an EMBL/Genbank/DDBJ accession number. Which is the case, and which file or database to use, should be explained in accompanying information.

<source>

The source of this feature. This field will normally be used to indicate the program making the prediction, or if it comes from public database annotation, or is experimentally verified, etc.

<feature>

The feature type name. We hope to suggest a standard set of features, to facilitate import/export, comparison etc. Of course, people are free to define new ones as needed. For example, Genie splice detectors account for a region of DNA, and multiple detectors may be available for the same site, as shown above.

<start>, <end>

Integers. <start> must be less than or equal to <end>. Sequence numbering starts at 1, so these numbers should be between 1 and the length of the relevant sequence, inclusive.

<score>

A floating point value. When there is no score (i.e. for a sensor that just records the possible presence of a signal, as "splice5" above) you must give something, by convention 0.

<strand>

One of '+', '-' or '.'. '.' should be used when strand is not relevant, e.g. for dinucleotide repeats.

<frame>

One of '0', '1', '2' or '.'. '0' indicates that the specified region is in frame, i.e. that its first base corresponds to the first base of a codon. '1' indicates that there is one extra base, i.e. that the second base of the region corresponds to the first base of a codon, and '2' means that the third base of the region is the first base of a codon. If the strand is '-', then the first base of the region is value of <end>, because the corresponding coding region will run from <end> to <start> on the reverse strand.

[group]

An optional string-valued field that can be used as a name to group together a set of records. Typical uses might be to group the introns and exons in one gene

prediction (or experimentally verified gene structure), or to group multiple regions of match to another sequence, such as an EST or a protein. See below for examples.

All strings (i.e. values of the <seqname>, <feature> or <group> fields) should be under 256 characters long, and should not include whitespace. The whole line should be under 32k long. A character limit is not very desirable, but helps write parsers in some languages. The slightly silly 32k limit is to allow plenty of space for comments/extra data.

Fields must be separated by TAB characters ('\t').

Here are some example records:

SEQ1	EMBL	atg	103	105	0.9	+	0
SEQ1	EMBL	exon	103	172	6.42	+	0
SEQ1	EMBL	splice5	172	173	0	+	.
SEQ1	netgene	splice5	172	173	0.94	+	.
SEQ1	genie	sp5-20	163	182	2.3	+	.
SEQ1	genie	sp5-10	168	177	2.1	+	.
SEQ2	grail	ATG	17	19	2.1	-	0

APPENDIX B

User Manual

1. Compiling And Running The Software

The software developed in this thesis work may be downloaded from http://iubio.bio.indiana.edu:7780/archive/00000645/. To begin using the software, a user would download the zip file, GFPE.zip, and then unzip the software to a selected directory on the hard drive. In our examples, this directory is named “gui”. After unzipping, the software files and directories will be shown as displayed in Figure 3.1. In the root directory of gui, this software package contains the graphical user interface Java files and an executable FFG program seen on the right side of Figure 1. and a GFPE directory that contains the Java files for the gene-finding evaluation programs, a data directory that can store gene sequence, annotation and prediction files, as well as the prediction result evaluation accuracy files and some temporary files, and a readme directory that contains the user manual image pages, all seen on the left side of Figure 1.

To run this tool, JDK, preferably version 1.4 or higher, should be installed on the machine. To compile, the command “javac *.java” should be used in the gui directory and in the GFPE directory separately. To run the tool, the command “java Gui” should be used, and also the correct class path should be set up in the system environment variables. For Unix systems, the user may need to set the “DISPLAY” variable to an appropriate value.

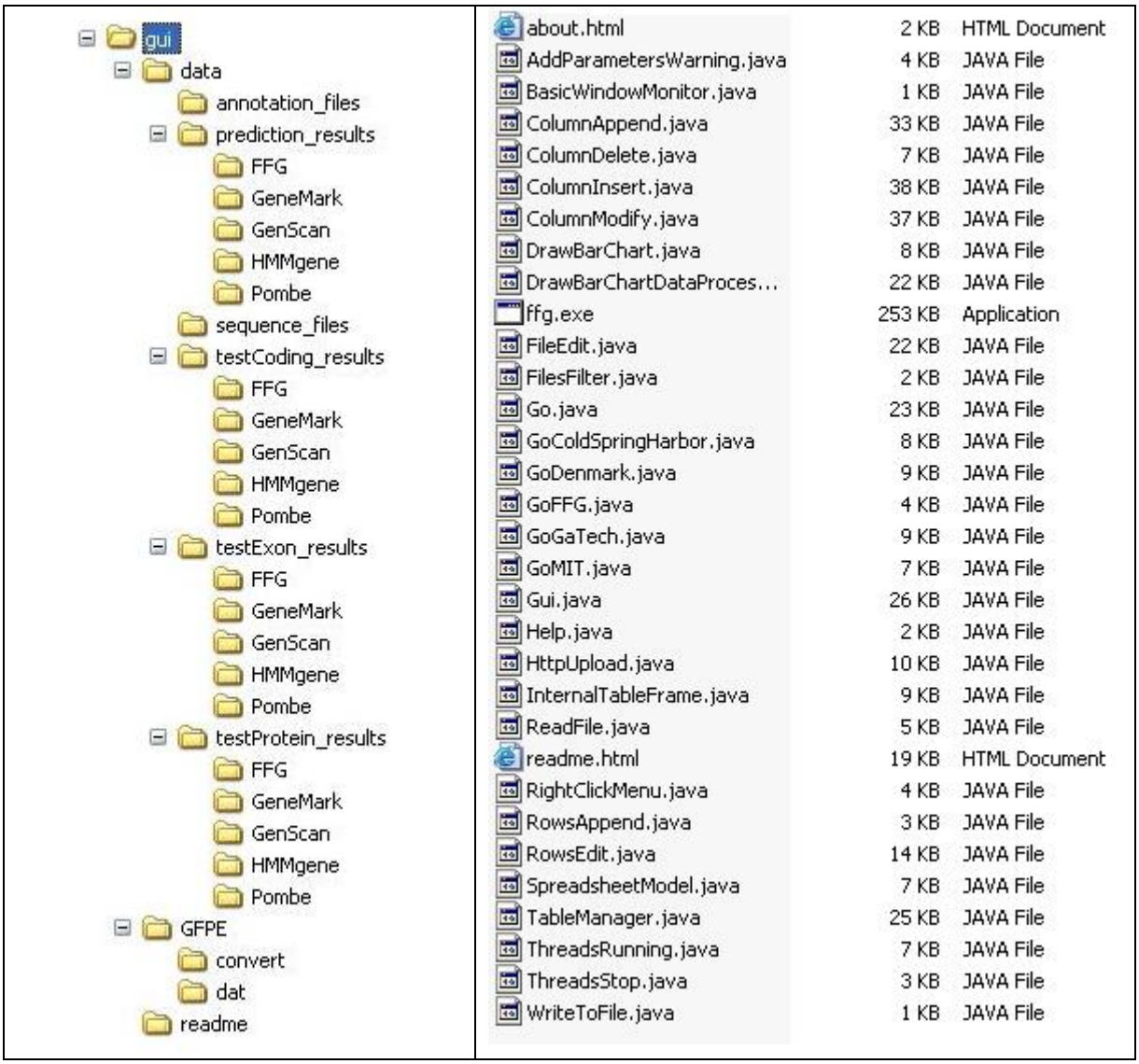

Fig 1 Directories of the software package and java files in the hard drive

2. Layout Of The Main Screen

Once the program is started, the main screen appears as seen in Figure 2. This initial screen contains the following parts:

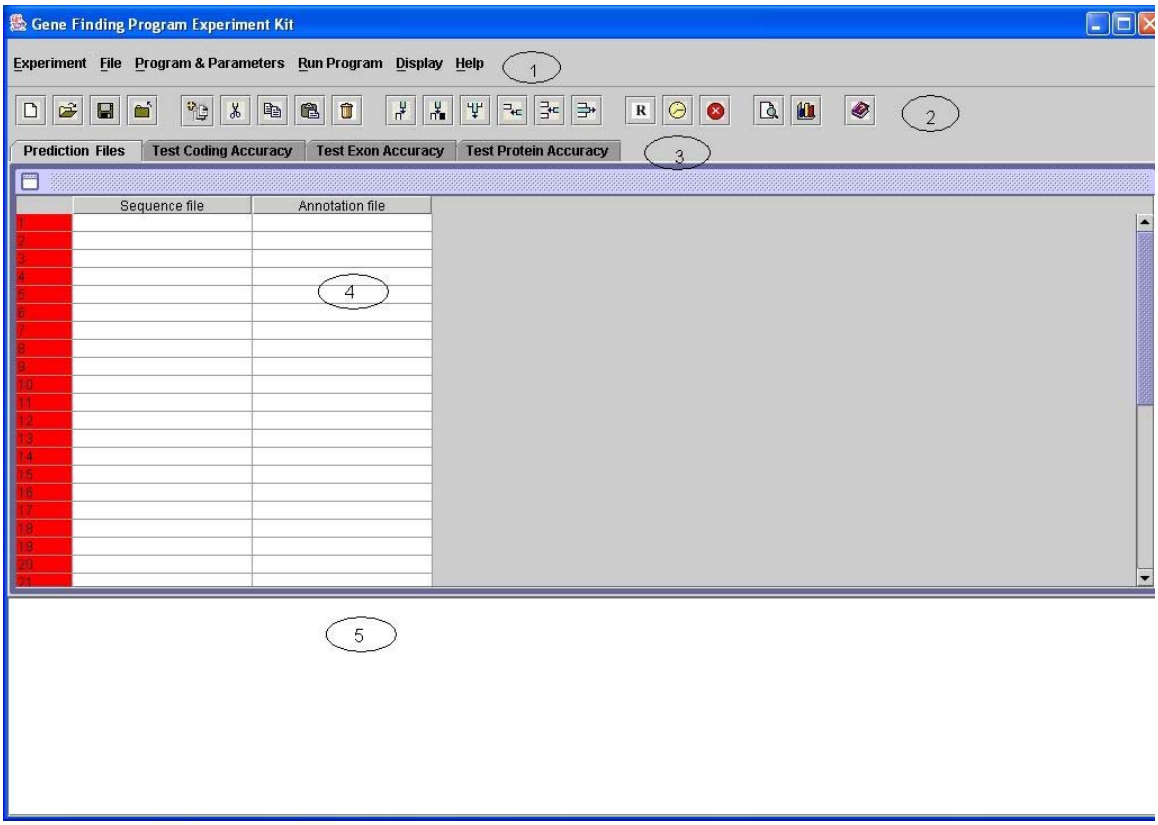

Fig 2 Layout of the main screen

- A JMenuBar on the top of the screen contains the menu items “Experiment”, “File”, “Program & Parameters”, “Run Program”, “Display” and “Help”. The detailed JMenuItems in each JMenu are as shown in Figure 3.
- A JtabbedPane in the middle of the screen contains four tabs: “Prediction Files”, “Test Coding Accuracy", "Test Exon Accuracy" and "Test Protein Accuracy".
- A spreadsheet-like table is attached to each tab. The table with the “Prediction Files” tab displays the DNA sequence files, annotation files and gene-finding program prediction files. The table with the “Test Coding Accuracy” tab displays the DNA sequence files, annotation files and gene-finding evaluation result files at the coding level. The table with the “Test Exon Accuracy” tab displays the DNA sequence files, annotation files and gene-finding evaluation result files at the exon level; The table with the “Test Protein Accuracy” tab displays the DNA sequence files, annotation files and gene-finding evaluation result files at the protein level. The first and second columns holds the same DNA sequence files and corresponding annotation files in all four tables.
- A text area on the bottom of the screen is used to display any file/files the user selects from the table, or program parameters of the user selected columns.

Experiment	File	Program & Parameters	Run Program	Display	Help
New	Add	Append Program	Run Prediction	Display Parameters	User Manual
Load	Cut	Insert Program	Run Evaluation	Display Files	About the Kit
Close	Copy	Modify Program	Show Running Programs	Draw Bar Chart	
Print	Paste	Delete Program	Stop Running		
Save	Delete	Append Blank Rows			
Save as		Insert Blank Rows			
Exit		Delete Rows			

Fig 3 GFPE Menu and Menu Items

3. “Experiment” Menu

- **New:** Open a new experiment. The new experiment is like the one shown in Figure 1, in which each blank table contains only two columns of sequence file and annotation file.
- **Load:** Open a saved experiment. The tables will contains all the saved files and related program columns.
- **Close:** Close the experiment. Ask “Do you want to save the change you made?” before close.
- **Print:** Not implemented yet.
- **Save:** Save the current experiment.
- **Save as:** Save the current experiment in the other name.
- **Exit:** Exit from the system.

4. “File” Menu

This menu is designed to operate on the first and the second column in four tables, “Delete” can work on all the columns.

- **Add:** Add files to a single column at each time. If the single column is the first column or the second column, the first column or the second column in all the four tables will contain the same sequence files or annotation files, even the user add the files only to one of the four tables. To add files, select appropriate table by clicking the tab using the mouse, and then select a single table cell as a starting cell to which the user wants to add files, or select an area, click “File” menu, click “Add”, a file selection window will popup, the user can select one or more files at each time, and then click open, the files will be added to the column where the selected single cell is or the first column of the selected area in the order they are selected (Figure 4).
- **Cut:** Cut files from the first or the second column, or both columns of the table only. Because all the files in other columns are created by different gene-finding programs based on the sequence file and annotation file in the first and the second column at the same row. If any of the first column or second column were cut, the files in other columns will be gone. To cut, select a single file or an area, click “File” menu, click “Cut”. Cut files are stored in a file named “tempDataStore” for future paste use, every new “Cut” will update this file.
- **Copy:** Copy files from the first or the second column, or both column of the table only. To make a copy, select a single file or an area within the first two columns, click “File” menu, click “Copy”. Copied files are stored in a file named “tempDataStore” for future paste use, every new “Copy” will update this file.
- **Paste:** Paste the cut or copied files, which are stored in tempDataStore, to the first or the second, or both column of the table only. To make a paste, select an equal size area within the first two columns, click “File” menu, click “Paste”. For the “Prediction Files Table”, the table cells that are at the same row as the pasted files and whose column number greater than 2 will become blank. For other evaluation accuracy tables, the corresponding table cells will be written by the words of “no pred result yet”.

- **Delete:** Delete the selected files from any of the four tables. To make a delete, select the files to be deleted, click “File” menu, click “Delete”.

The user also can right click mouse button to get fast access to the above menu item function after selecting table cells.

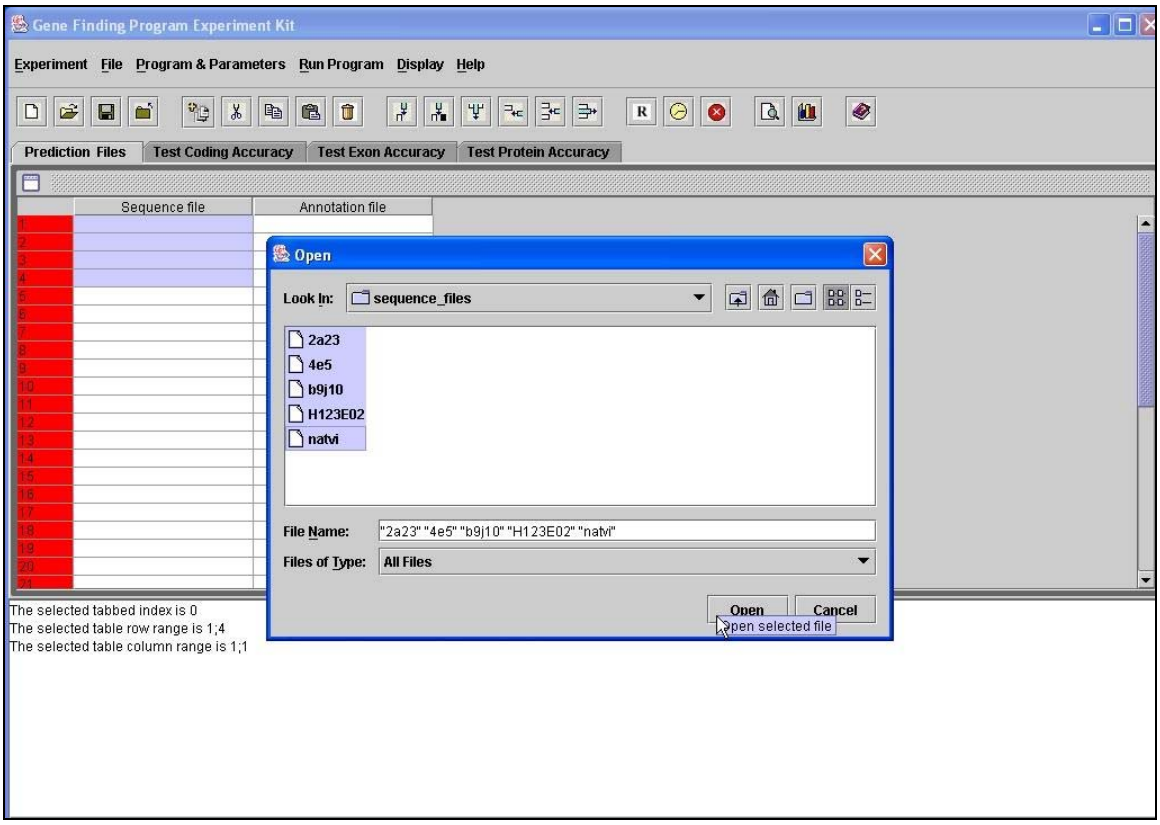

Fig 4 Add files to the “Prediction Files” Table

5. “Program & Parameters” Menu

This menu is designed to assist user append, or insert gene-finding programs, or blank rows to the table, change program parameters, or delete unwanted programs or unwanted rows from the table. There are five gene-finding programs in this package, they are: GenScan, HMMGene, GeneMark, Pombe and FFG program. Please perform the following functions on the “Prediction Files” table. Once each operation is done, all the other three tables will do the same.

- **Append Program:** Append a user selected program to the end of each table. To append, click “Program & Parameters” menu, click “Append Program Column”, a “Select Program and Setup Parameters Window” will popup, as shown in Figure 5a, 5b, 5c, 5d, and 5e. The user can select a program from one of five programs in the drop down JComboBox and then setup its parameters.
- **Insert Program:** Insert a user selected program to the left hand side of the selected column in each table. Insertion to the left hand side of the first or the second column is not allowed. To insert, select a column located, click “Program

& Parameters" menu, click "Insert Program Column". Then the user can select a program from the popup window and then setup its parameters.

- **Modify Program:** Change program parameters or using a new program to replace the original selected program in the selected column. To modify, selected a column to be changed, click "Program & Parameters" menu, click "Modify Program Column", or click the to be changed column head directly. The user re-setup the parameters without changing the program in the popup window, or select a new program to replace the original program and then setup new program parameters. Figure 6 shows a table setup with five selected programs.
- **Delete Program:** Delete a program from each table. Pay attention to do this, all the data and the column can not be recovered anymore once the deletion is complete. To delete, click "Program & Parameters" menu, click "Delete Program Column".
- **Append Blank Rows:** Append a number of blank rows to the end of each table. To append, click "Program & Parameters" menu, click "Append Blank Rows". A popup window will ask how many rows to be added, input a number, then click ok button.
- **Insert Blank Rows:** Insert a number of blank rows before the selected row in each table. To insert, click "Program & Parameters" menu, click "Insert Blank Rows". A popup window will ask how many rows to be inserted, input a number, then click ok button.
- **Delete Rows:** Delete a number of selected rows from each table. Pay attention to do this, all the data and the rows can not be recovered anymore once the deletion is complete. To delete, select the rows to be deleted, click "Program & Parameters" menu, click "Delete Rows".

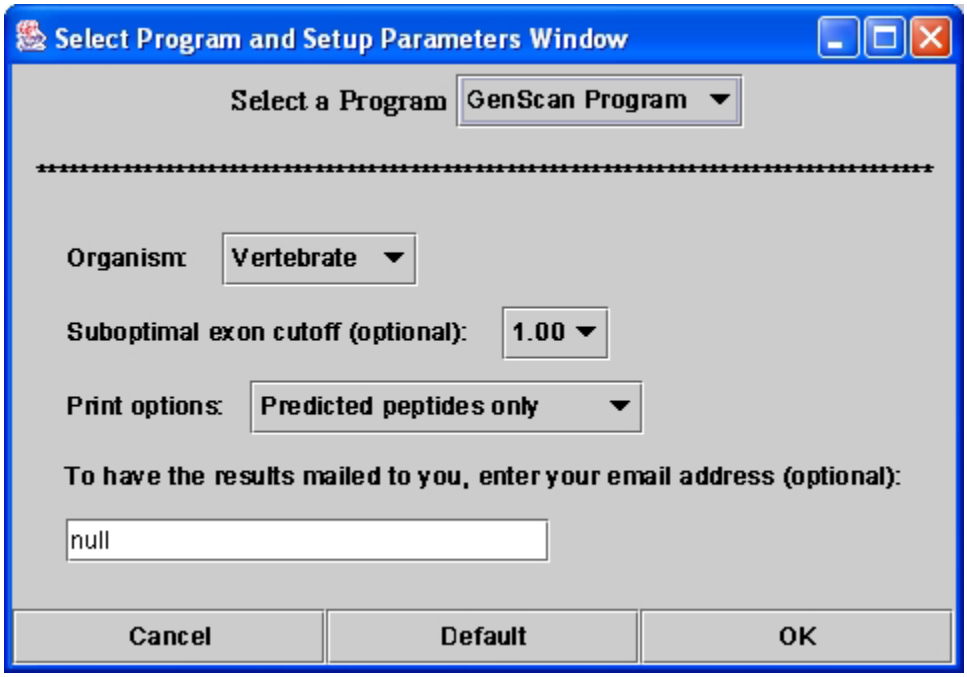

Fig 5a Select Program and Setup Parameters Window for GenScan Program

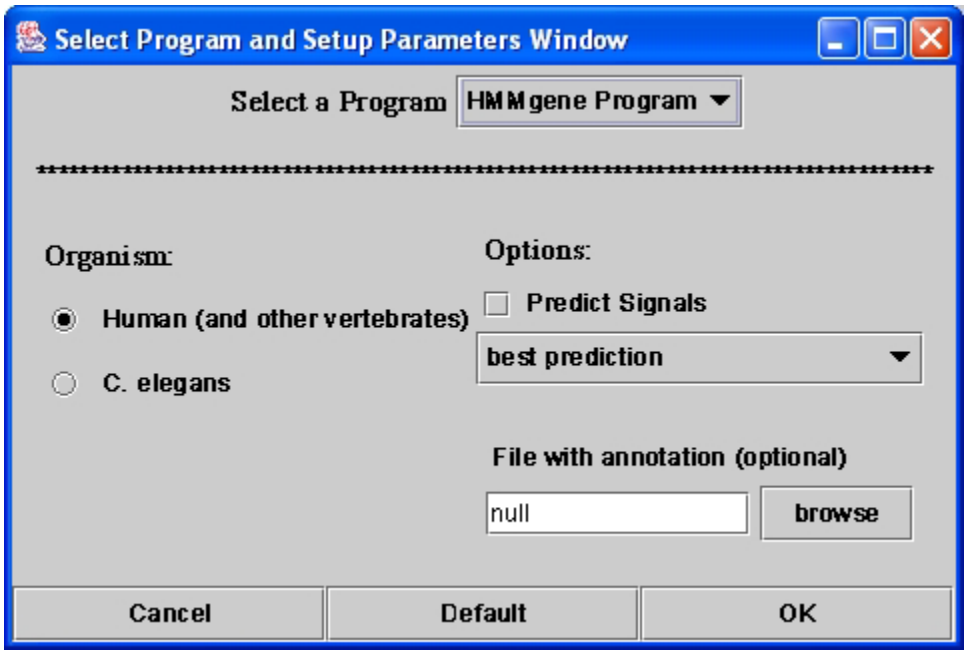

Fig 5b Select Program and Setup Parameters Window for HMMGene Program

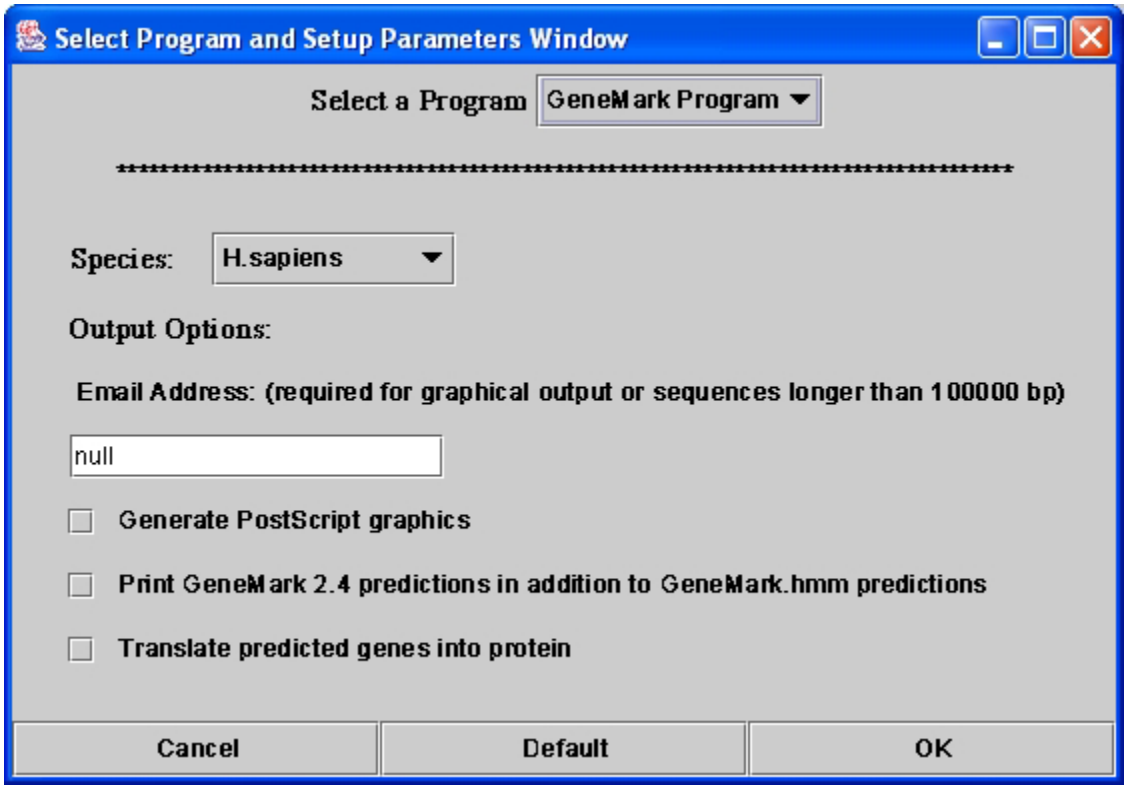

Fig 5c Select Program and Setup Parameters Window for GeneMark Program

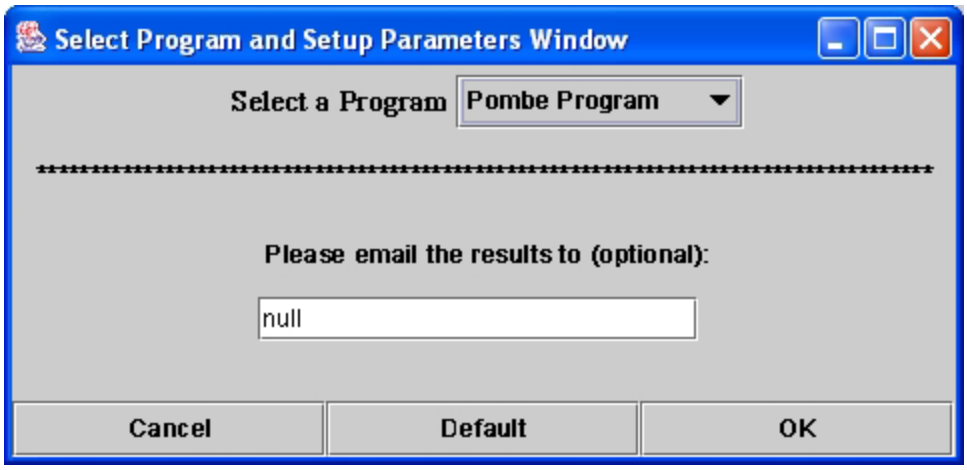

Fig 5d Select Program and Setup Parameters Window for Pombe Program

Fig 5e Select Program and Setup Parameters Window for FFG Program

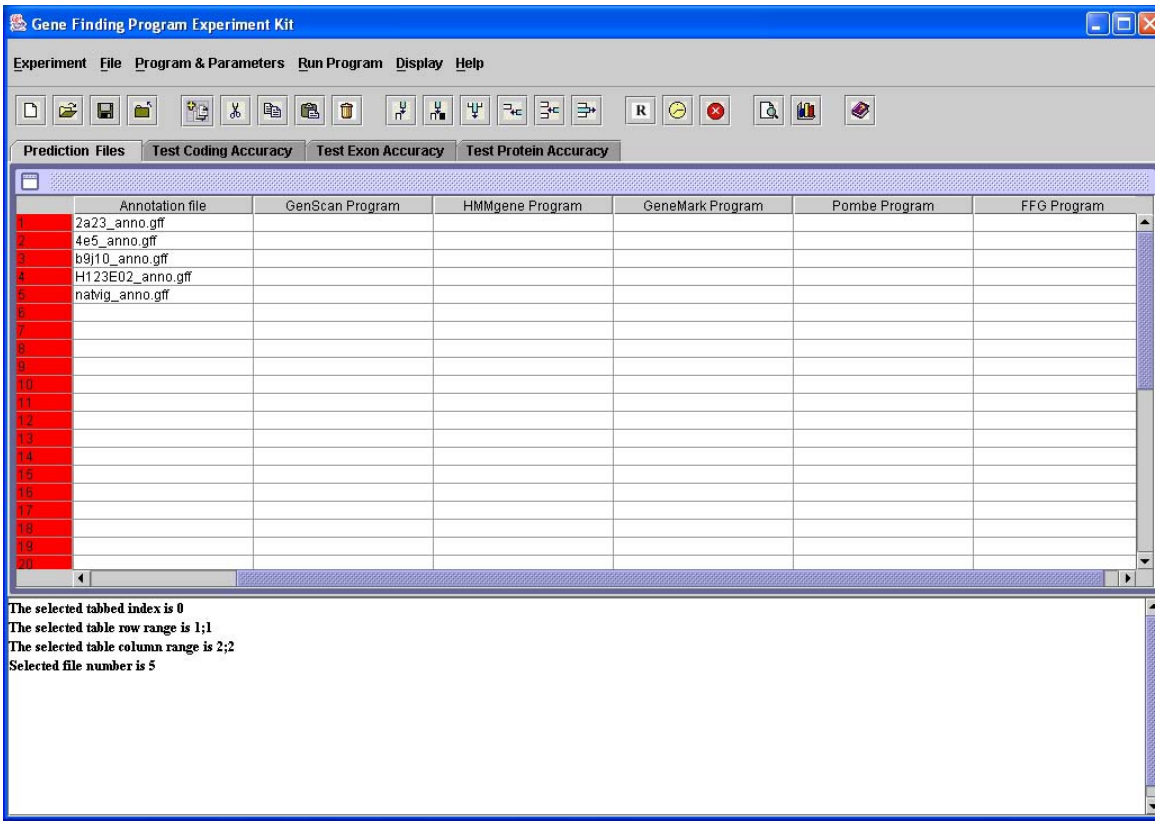

Fig 6 A table setup with five gene-finding programs

6. “Run Program” Menu

This menu is designed to assist user run the gene-finding programs in the first table - Prediction Files table, run test coding accuracy in the second table – Test Coding Accuracy table, run test exon accuracy in the third table – Test Exon Accuracy table, run test protein accuracy in the fourth table – Test Protein Accuracy table. The data flow through the GFPE graphical user interface is shown in Figure 7. The Run Prediction should go first, since the evaluation needs the prediction data.

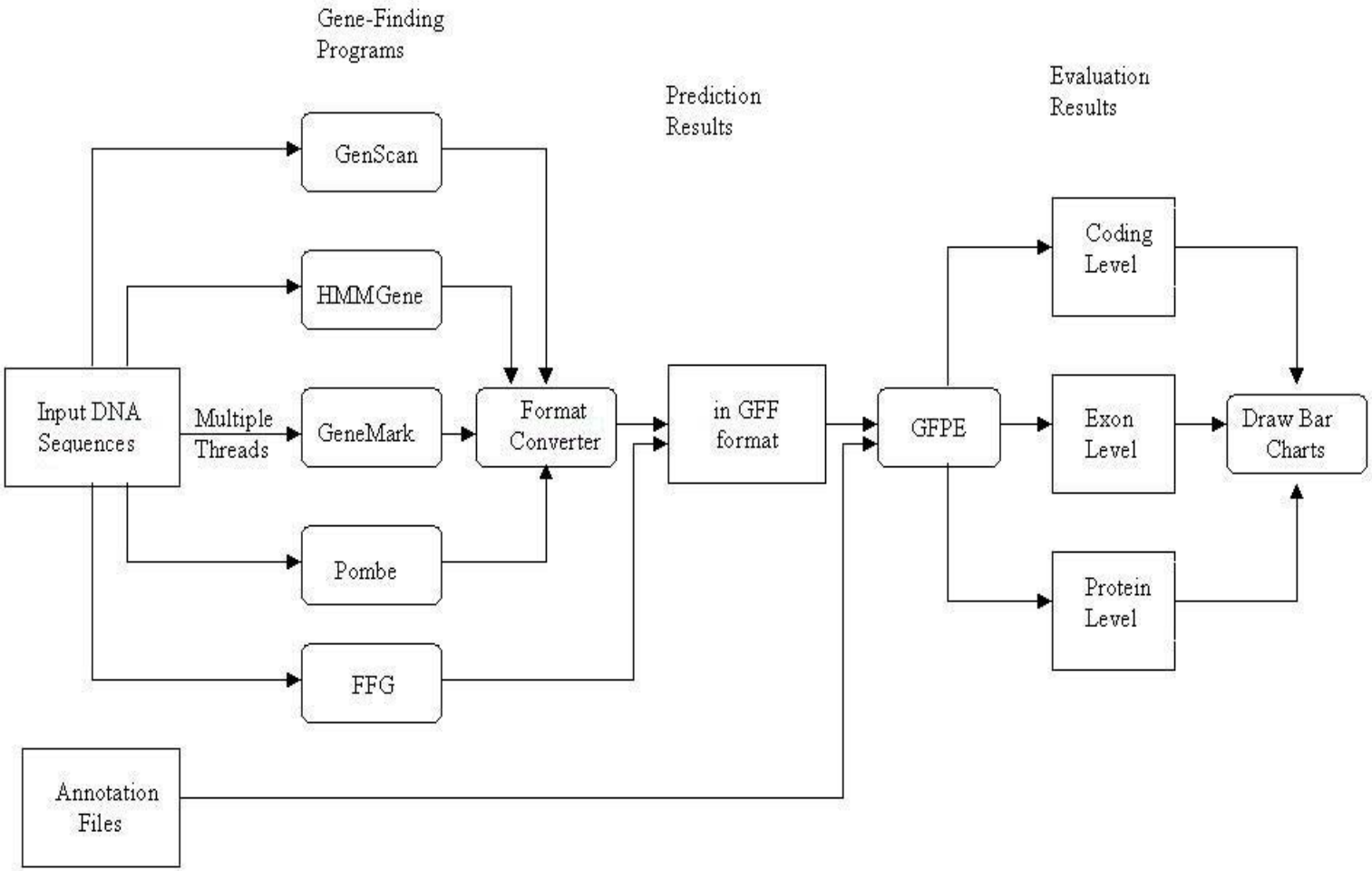

Fig 7 Data flow through the GFPE Graphical user interface

- **Run Prediction:** Run the user selected programs using multiple threads in the first table. To run, select an area in the table, click “Run Program” menu, click “Run Prediction”. When the prediction result is available, the system will write the result file to the appropriate prediction_results folder on the hard drive, see Figure 1, and write the file name to the appropriate table cell in the table, update the corresponding table cells in the other three tables using the words of “not evaluated yet”. User can monitor the run prediction process by clicking “Show Running Programs”.
- **Run Evaluation**: Run the evaluation programs of test coding accuracy, test exon accuracy and test protein accuracy by selecting the files to be evaluated in the second, the third and the fourth table, respectively. To run, go to appropriate table, select an area, click “Run Program” menu, click “Run Evaluation”. The system will automatically recognize what evaluation program (test coding, test exon or test protein) to execute based on the selected table. When the evaluation result is available, the system will write the result file to the appropriate folder on the hard drive, and update the table cell using the file name.
- **Show Running Programs:** Show the unfinished running programs. To show running programs, during the run prediction process, click “Run Program” menu, click “Show Running Programs”, see Figure 8. The show window will be automatically updated by the latest unfinished running program information every 10 seconds. After all the programs finished, it shows “All done!”.
- **Stop Running:** Stop the running programs. To stop, click “Run Program” menu, click “Stop Running”, which will activate the stop points in multiple thread programs, block the result files to be written to the file storage folders on the hard drive and to the table cells.

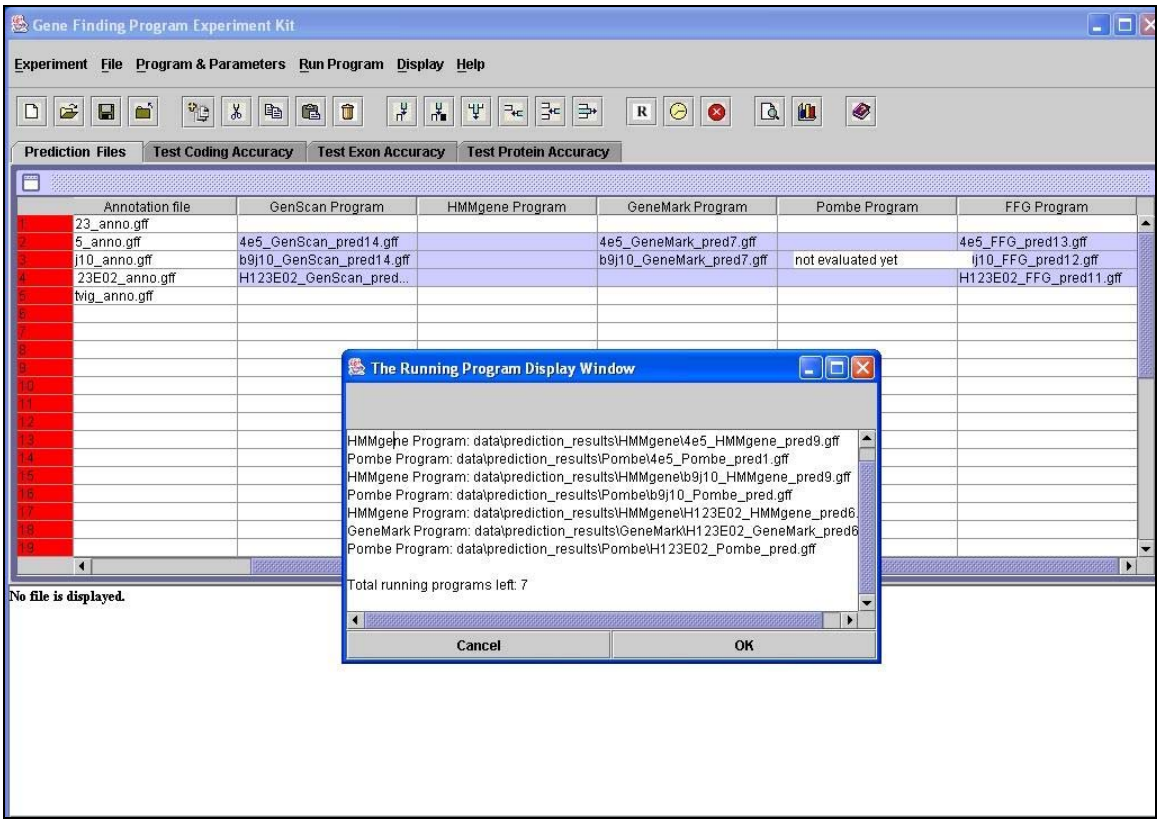

Fig 8 Monitor running program window

7. "Display" Menu

This menu is designed to assist user check parameter, or file information through the bottom text window, and compare the prediction accuracy results using the weighted average values shown in bar charts.

- **Display Parameters:** Display the parameters for the selected gene-finding program. To display, go to the first page table, select the column, click "Display" menu, click "Display Parameters", the parameters will be shown in the bottom text window. Or click the column head, check the parameters from the popup window.
- **Display Files:** Display the selected file content. To display, go to appropriate table, select an area that contains the files to be checked, click "Display" menu, click "Display Files", then the selected files will be displayed one by one in the bottom text window.
- **Draw Bar Chart:** Draw coding, exon or protein level accuracy bar chart based on the user selected evaluation data. The data will be averaged by weight if multiple sequence evaluation results are selected in one column. User also can compare single sequence prediction evaluation using the same gene-finding program. To draw the bar chart, go to appropriate evaluation table, select an area that contains the files to be compared, click "Display" menu, click "Draw Bar Chart", a popup window will display the bar chart, see Figure 9, 10 and 11.

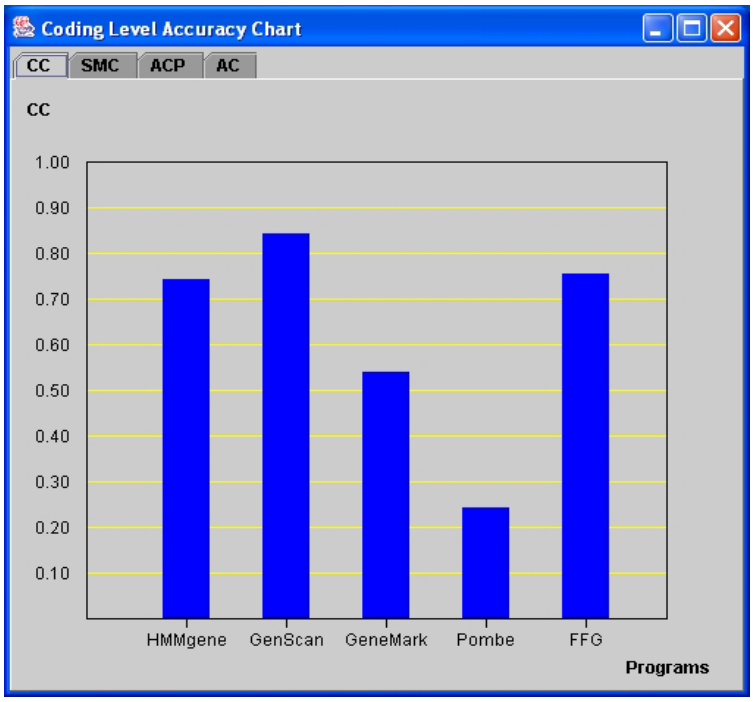

Fig 9 Draw bar cart – coding level accuracy

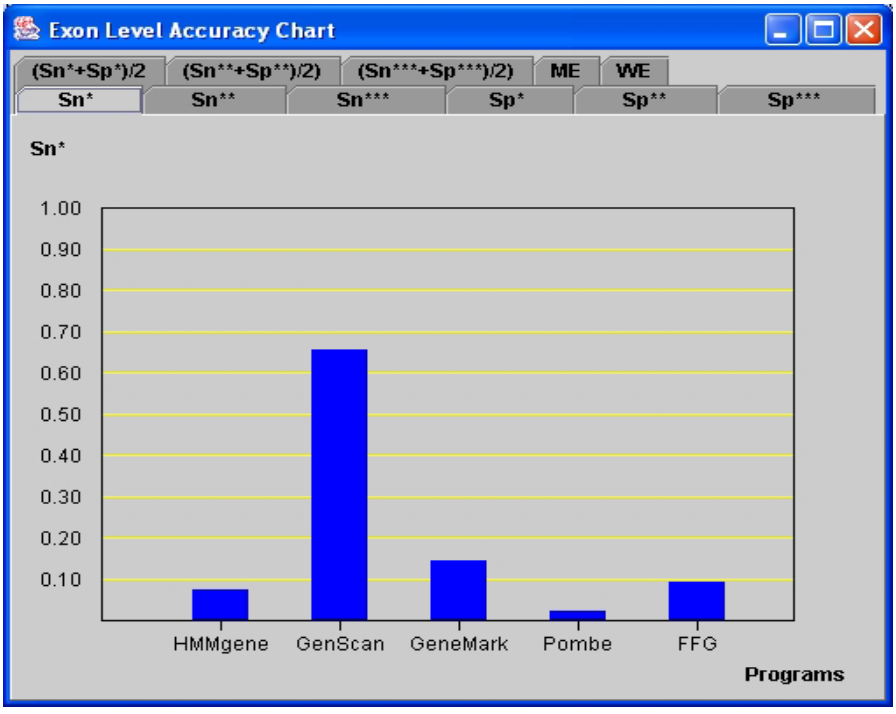

Fig 10 Draw bar cart – exon level accuracy

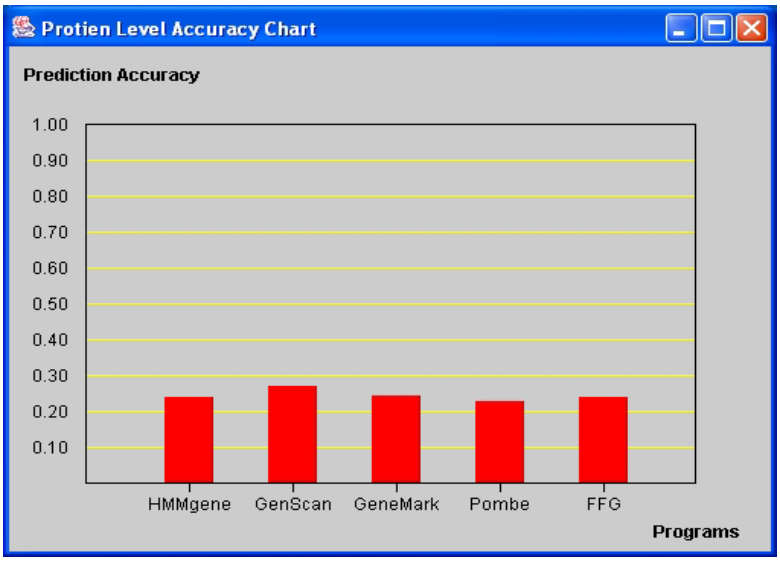

Fig 11 Draw bar cart – protein level accuracy

8. “Help” Menu

This menu is designed to assist user know how to use the GFPE graphical user interface kit and the version of this software package.

- **User Manual:** Describe the user manual. To use, click “Help” menu, click “User Manual”.
- **About the Kit:** Describe version, developer and the contact information about the GFPE graphical user interface kit.

APPENDIX C

How To Add a New Gene-Finding Program Into the GFPE Package?

Follow the procedures below to add a new gene-finding program into the GFPE package step by step, suppose the new program name is AAA and it's on a remote server.

1) Run AAA program on the remote server: Write a basic java class file GoAAA.java, make upload DNA sequence file, upload the related parameters to the remote server, get the prediction result back and change it into the standard GFF format success. Refer to the class GoMIT.java and GoDenmark.java.
2) Setup result file related storage directories, output file names and paths: On the local hard drive, make four AAA folders in the directories of "data\prediction_results\", "data\testCoding_results\", "data\testExon_results\" and "data\testProtien_results\" to hold the AAA program prediction result, coding level test result, exon level test result and protein level test result, respectively. In the class Go.java, add "AAA Program" to the end of the field String array "programs"; add "data\\prediction_results\\AAA\\" to the end of the field String array "predictionResult_filePath" to setup the prediction output result file path; add "_AAA_pred.gff " to the end of the field String array "predictionResult_suffix" to setup the prediction file name suffix; add "data\\testCoding_results\\AAA\\" to the end of the field String array "testCodingResult_filePath" to setup the coding level test output result file path; add "data\\testExon_results\\AAA\\" to the end of the field String array "testExonResult_filePath" to setup the exon level test output result file path; add "data\\testProtein_results\\AAA\\" to the end of the field String array "testProteinResult_filePath to setup the protein level test result file path. Refer to the class Go.java.
3) Code parameter related panel and methods in the class ColumnInsert.java, ColumnModify.java and ColumnAppend.java: Add "AAA Program" to the end of the field String array "programs" to allow user choose the new program; code an AAA program parameter panel; update the methods "MyItemListener{}", "saveProgramPramaters()" and "saveProgramDefaultParamaters()"; code the new methods getAAA_Parameters() and getDefaultAAA_Parameters().
4) In the class Go.java, find the method "public void runPrediction ()", add a boolean judgement to determine when to run AAA program. Refer to the class Go.java.
5) Modify the file GoAAA.java: Make it be able to write the prediction result into a file on the local disk, and display the file properly in the table.

For a gene-finding program that can be downloaded to the local machine, refer to the above procedures and the class GoFFG.java and Go.java to add it into the GFPE package.

REFERENCES

1. Borodovskky M. and McIninch J. GeneMark: parallel gene recognition for both DNA strands. *Comput. Chem*. 1993, 17: 123-133.
2. Burge C. and Karlin S. Prediction of complete gene structure in human genomic DNA. *J. Mol. Biol.,* 1997, 268:78-94.
3. Burset, M.; Guigo, R. Evaluation of gene structure prediction programs. *Genomics*, 1996, 34:353-367.
4. Chen T. and Zhang M. Q. Pombe: a gene-finding and exon intron structure prediction system for fission yeast. *Yeast*, 1998, 14:701-710.
5. Edelmann S. E. and Staben C. A statistical analysis of sequence features within genes from *Neurosora crassa*. *Exp. Mycol.,* 1994, 18:70-81.
6. Java Tutorial: http://java.sun.com/docs/books/tutorial/
7. Kraemer E.; Wang J.; Guo J.; Hopkins S.; Arnold J. An analysis of gene-finding programs for Neurospora crassa. *Bioinformatics*, 2001, 17(10): 901-912.
8. Krogh A. Two methods for improving performance of an HMM and their application for gene finding. In *Proceedings of Fifth International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA. 1997, pp.179-186
9. Lukashin A. V. and Borodovsky M. GeneMark.hmm: new solutions for gene finding, *Nucleic Acids Research*, 1998, 26:1107-1115.
10. Malacinski G. M., Freifelder D. *Essentials of Molecular Biology*, 3rd edition, Jones and Bartlett Publishers, Inc. 2002, 325-326.

11. Sanger Center: GFF (2000). GFF (Gene Finding Features) Specifications Document: http://search.cpan.org/src/LDS/AcePerl-1.83/docs/GFF_Spec.html

12. Wang J.; Kraemer E. GFPE: Gene-finding Program Evaluation. *Bioinformatics*, 2003, to appear.