Fall 2007

A quantitative study of gene identification techniques based on evolutionary rationales

Cyriac Kandoth

A QUANTITATIVE STUDY OF GENE IDENTIFICATION TECHNIQUES BASED ON EVOLUTIONARY RATIONALES

by

CYRIAC KANDOTH

A THESIS

Presented to the Faculty of the Graduate School of the

Missouri University of Science and Technology

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2007

Approved by

Dr. Fikret Ercal, co-Advisor

Dr. Ronald L Frank, co-Advisor

Dr. Jennifer Leopold

ABSTRACT

Current gene identification (GI) techniques typically rely on matching biological or chemical properties of specific genes, specific species, specific ecotypes, etc. Other techniques might involve homology searches using known gene sequences. Since they are either too specific or they depend on known genes, these techniques can never claim to be complete i.e. to have identified all possible genes in a genome. This is an inherent drawback caused by the immense complexity of gene organization. However, it is possible to get closer to a more global generalized GI technique by using evolutionary rationales. The advantage of such a general technique is that, once automated on a computer, it can be easily extended to identify any gene that evolved with that rationale. In this thesis, a new automated GI technique is proposed, and compared against another computer-based technique proposed earlier. Both methods utilize EST data available from NCBI databases to discover previously unknown genes. The newly proposed method identifies one gene family at a time and is based on a distinctive negative selection pattern (NSP) of differences, which is seen between the coding regions of gene family members. The other technique, called ESTminer, attempts genome-wide gene family identification for any organism, by detecting single nucleotide polymorphisms between potential family members. In this thesis, a complete automated analysis of both techniques is presented.

ACKNOWLEDGEMENTS

I am extremely grateful to the many people who have contributed, even in little ways, to the development of this thesis. But first, I have to thank both Dr. Fikret Ercal and Dr. Ronald L Frank - Dr. Ercal, for his invaluable guidance into the academic realm, and Dr. Frank, for sharing his immense "repository of information". I am also grateful to Dr. Jennifer Leopold for the valuable feedback she provided as a member of the examining committee.

I will not forget the many moments spared by the faculty members and graduate students at the Department of Computer Science. They have all contributed to this thesis in many little ways, whether they know it or not.

Finally, I am in debt to my friends and family for their continuous support throughout the development of this thesis.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Gene Identification (GI) is the process of finding segments within genomic data (like DNA sequences) that contribute a specific functionality i.e. a gene. Today, there are hundreds of very different GI techniques. These techniques can be specific to plants, specific to mammals, specific to certain species or ecotypes, specific to a gene family, or sometimes even gene-specific. This is because the techniques usually depend on one or more biological properties of genes that make it possible to pin-point them within a sea of DNA sequences. Also, these techniques are often conducted experimentally which makes them slow and tedious. This is why there is a move toward developing automated GI techniques. "Automated", in this context, refers to using a computer to analyze raw genomic data and produce ready and conclusive information for a biologist. A review of publications in GI found that very few attempts were made to create a fully automated general process for identification of genes throughout a genome, or at least throughout a gene family. This might imply that the particular problem of developing such a large scale non-specific GI technique is either very difficult or, considering the complexity of gene organization, maybe even impossible. However, Bioinformatics - the application of computers to solve biology problems - is still a fledgling field and there is plenty of scope for new ideas.

In this thesis, the recent work that has gone into computer-based GI processes is first examined in Chapter 1. One of these processes - called ESTminer - claims to identify potential gene families within an entire genome. This is tested thoroughly in Chapter 2. ESTminer was developed by Nelson et al. in 2005. Chapter 3 discusses another GI technique which uses negative selection patterns (NSP) between gene family members to identify all members of that gene family. This process was first developed and automated by Frank et al. in 2006. The chapter goes on to explain how this automation was further developed using perl scripts that could interface with online applications such as BLAST (Basic Local Alignment Search Tool) and ORF Finder (Open Reading Frame Finder). The only input that this automation needs is a known gene (a protein sequence) that belongs to the gene family to be identified. The output is a table that summarizes the distribution of negative selection patterns (NSP) between contigs (a contiguous set of

overlapping sequences which could potentially represent a gene) from that family. This information helps to identify potential members of the same gene family as the protein-coding gene used as input. The correctness of this automation is validated using sample sequences from *Arabidopsis thaliana* (abbreviated "At") to identify a previously known gene family.

1.1. GENE IDENTIFICATION

Automated analysis of genomic data, using techniques developed for bioinformatics, came about as a result of necessity. Genomic sequences are enormous and manual analysis is impractical. The progress of the Human Genome Project is a good example (Human Genome Program, 1994). When it started, the identification of genes was a slow and tedious process. It usually involved matching known genes from other species with those in the human genome. By 2003, the entire 3 billion nucleotides were sequenced, but the processes used in locating the genes became numerous and elaborate. Some of these were conducted experimentally (in a laboratory) while most were conducted "in-silico" (on a computer) because of the enormity of the genomic data. However, most of these techniques were based on biological or chemical properties that were too specific. These limitations led us to look for a more general non-specific technique that made use of the high resolution DNA sequences from various genomes stored at enormous public-access databases, in particular, the databases at NCBI (National Center for Biotechnology Information). NCBI BLAST is a publicly accessible online application which searches through these databases for DNA sequences which are similar to a given query sequence.

In order to find the general rules by which DNA sequences have evolved, and subsequently apply them in a gene identification technique, the best option was to use evolutionary rationales. Unfortunately, this only widens the generalization because different classes of organisms have evolved very differently and developed their own evolutionary mechanisms. So, a technique based on evolutionary rationales is not entirely universal. For instance, most plants have evolved using the same common mechanisms and several global gene identification techniques based only on these mechanisms, can be formulated. But such a technique cannot be directly extended for mammal genomes since

they have evolved very differently from plants. In this thesis, the two techniques described are designed specifically for plant species and tested on *Arabidopsis thaliana*.

1.2. RELATED WORK

Before getting to the two GI techniques that this thesis focuses on, some related techniques of gene identification are reviewed. Bie et al. presented CAFÉ (Computational Analysis of Gene Family Evolution) for analyzing and predicting the evolution of the size of gene families in a phylogenetic context i.e. pertaining to the evolutionary history of a particular group of organisms (Bie, Cristianini, et al., 2006). This method modeled gene gain and loss along each lineage of a phylogenetic tree using a random birth and death process, and then used that model to calculate the probability of transitions in gene family size from parent to child node in the phylogeny. Given a gene family and its evolutionary analysis, DETECTER (Determining Clinically relevant Transmutations using Evolutionary Rationales) was designed to predict sites in a protein sequence where amino acid replacements are likely to have a significant effect on phenotype, including causing genetic diseases (Gaucher De, et al., 2006).

Hekmat-Scafe et al. (2002) presented their methodology for identifying multiple potential odorant-binding protein (OBP) family members through a PSI-BLAST (Position Specific Iterative BLAST) search of Drosophila genomic sequences at NCBI, in particular the olfactory-specific OS-E protein sequences. The resulting sequences are used to scan Drosophila genomic sequences at NCBI using TBLASTN (a version of BLAST that takes a protein query and returns similar sequences from the NCBI nucleotide databases), generating more OBP-like products. Phylogenetic analysis is then applied to remove the identified genes, and scan the Drosophila genome using the remaining sequences. Tian et al. developed a strategy to identify 57 and 32 GRAS gene family members in rice and Arabidopsis respectively (Tian et al., 2004). The method starts with a single sequence as a query to search through multiple rice genome databases using TBLASTN. GRAS genes in Arabidopsis were identified with BLASTP (version of BLAST that takes a protein query and returns similar sequences from the NCBI protein databases) and aligned using ClustalX (a multiple sequence alignment tool). Phylogenetic trees were constructed using ClustalX, MEGA2 (Manipulation Environment of Genetic Analyses), and PHYLIP (Phylogeny Inference Package); motifs (repeatedly occurring

sequence patterns) were identified using MEME (Multiple Em for Motif Elicitation); and divergence time was estimated using PAML (Phylogenetic Analysis by Maximum Likelihood). Nakano et al. identified 122 and 139 ERF family members in Arabidopsis and rice respectively, using gene structure analysis, comparative and phylogenetic analysis, and motif detection (Nakano, Suzuki, 2006). Liu identified 9 ACT domain repeat protein-coding genes based on similarity search and domain detection (Liu, 2006).

Other automated or semi-automated processes have also been developed for identifying gene families. Brown et al. developed a semi-automated method for mining ESTs (Expressed Sequence Tags - short nucleotide fragment sequences) for gene discovery and functional characterization in a major facilitator superfamily (MFS) of transporter genes (Brown et al., 2003). The strategy starts with a seed protein sequence, and collects a core family of related sequences by running PSI-BLAST. Then a collection of ESTs is generated by a TBLASTN search in the NCBI EST database (dbEST). After removing non-mammalian vector sequences and previously characterized ESTs, the remaining ESTs are assembled using CAP3 (a popular Contig Assembly Program). The generated contigs and singletons are candidates for new genes and are evaluated for membership with specific MFS families.

1.3. ESTMINER

ESTminer compares similar sequences throughout the genome of a specific ecotype (a subdivision of a species characterized by its ecological surroundings) and tries to find single nucleotide polymorphisms (SNPs) between them. An SNP, as the name suggests, is a variation in a single nucleotide base between two DNA sequences (Figure 1). When ESTminer finds certain SNPs between two otherwise very similar sequences

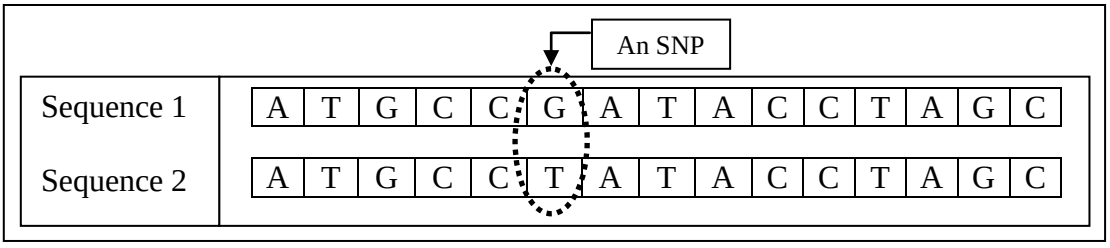

Figure 1.1 - A Single Nucleotide Polymorphism (SNP) between two similar sequences

(labeled Locus Defining Polymorphisms), it marks these sequences as possibly representing genes belonging to the same family. These sequences (which are usually ESTs) are referred to as potential Haplotypes (pHaps). Some of these pHaps are contigs assembled from the ESTs. In this study, these two types are differentiated as pHap ESTs and pHap contigs.

Before running the ESTminer suite of programs, ESTs of the same ecotype need to be collected and assembled using CAP3 with its parameters configured specifically for that ecotype. A database is created containing both the ESTs and the resulting contigs. Each contig is then submitted as a query to a BLAST search over this local database. Each query collects the ESTs and contigs that are similar to it. According to Nelson et al., this is equivalent to collecting all potential genes that belong to the gene family that each contig query might represent (Nelson et al., 2005). BLAST arranges these ESTs in order of quality of alignment. This makes it easy for ESTminer to later pick out the ESTs or contigs with locus defining polymorphisms and to designate them as pHaps.

1.4. NEGATIVE SELECTION PATTERNS TO IDENTIFY GENE FAMILIES

The evolutionary rationale for this technique is based on a specific negative selection pattern which is a result of gene duplication (when a gene is erroneously copied over twice in the same genome). Duplication allows the duplicate copies of a gene (also known as paralogs) to mutate freely without selective pressure and acquire new or altered functions while another copy retains the functions of the original gene. Susumu Ohno argues that gene duplication is the most important evolutionary force (Ohno S, 1970). Its status as the most common evolutionary mechanism in plants makes it a popular rationale to develop generalized gene family identification techniques. The technique proposed in this paper tries to find a characteristic pattern of nucleotide substitutions (mutations) between potential paralogs with respect to their position within a codon (a triplet of nucleotides that codes an amino acid). Each codon can be coded by the 4 different nucleotide bases - Adenine, Guanine, Cytosine, and Thymine. This allows 4^3 different types of triplets i.e. 64 different triplets out of which only 61 are codons (i.e. only 61 encode amino acids). However, some of these codons encode the same amino acids because they share a similar sequence of nucleotide bases. So, despite the 61 different codons, there are only 20 distinct amino acids. This redundancy allows certain single

nucleotide substitutions to occur, that change the codon, without changing the resulting amino acid. This is known as a synonymous substitution. The gene ends up producing the same protein as before and the mutation is carried over into future generations. Alternatively, a mutation that changes the codon to encode a different amino-acid is called a non-synonymous substitution. When two very similar sequences appear to have more synonymous differences between each other than non-synonymous ones, they could possibly be paralogs that diverged from each other after a gene duplication event. The level of divergence from each other can even be used to estimate when the divergence occurred.

In particular, single nucleotide substitutions in the third position of a codon almost always produce the same amino acid. Some first position substitutions also produce the same amino acid, but they are not as redundant as third position substitutions. Substitutions in the second position of a codon never produce the same amino acid. So, if differences between two paralogs are evolutionary and subject to negative selection, significantly more differences will occur in the third position and the least will occur in the second position. However, if differences between paralogs are artifacts (cDNA cloning, sequencing errors, etc.) then no pattern in codon positions should be exhibited.

Note that all members of a particular gene family need not be detectable by this technique. This is because negative selection is not the only evolutionary mechanism. Sometimes non-synonymous substitutions can turn out to be beneficial (positive or adaptive selection). Also, given time, paralogs could diverge so completely from each other that it would be impossible to know that they ever belonged to the same family.

2. ANALYSIS AND EVALUATION OF ESTMINER

In 2004, Nelson et al. released a suite of programs that attempted to perform gene and allele identification throughout the genome of an ecotype (Nelson et al., 2004). The only input that the programs require is a file containing all (or as many of) the known ESTs of that ecotype. In this thesis, their suite of programs is tested by running it on the Columbia ecotype of *Arabidopsis thaliana*. *Arabidopsis thaliana* was chosen since it is the only plant with its entire genome sequenced. The NCBI MapViewer contains the entire genome of *Arabidopsis thaliana*, with the locations of known genes mapped into it. The start and stop positions of these previously identified genes are provided in the NCBI MapViewer application which is accessible online and updated frequently.

2.1. OVERVIEW OF THE TECHNIQUE

According to Nelson et al., the correct operation of ESTminer is hugely dependent on the parameters used in the contig assembly step. As Dr. Nelson puts it –

"The optimum settings for CAP3 may need to be adjusted for each dataset. You must look at the CAP3 assemblies using your own EST data and see how changes to the -o (overlap length cutoff) and -y (clipping region) options affect its output. The objective is to choose values which produce contigs that represent closely related sequences without splitting groups inappropriately, but at the same time not including sequences which match the others by only a limited amount of sequence similarity."

(Personal communication, March 16 2007)

In other words, the number of contigs that CAP3 assembles should reflect the number of identifiable gene families within the given EST data set. Before running ESTminer, two input files needed to be generated - BlastDB and BlastOut. This was performed as described in Figure 2.

After running the ESTminer suite of programs on BlastDB and BlastOut, ESTminer's huge set of resulting pHaps needed to be analyzed. Three primary scripts were created to perform the analysis. It was necessary to run them one after another.

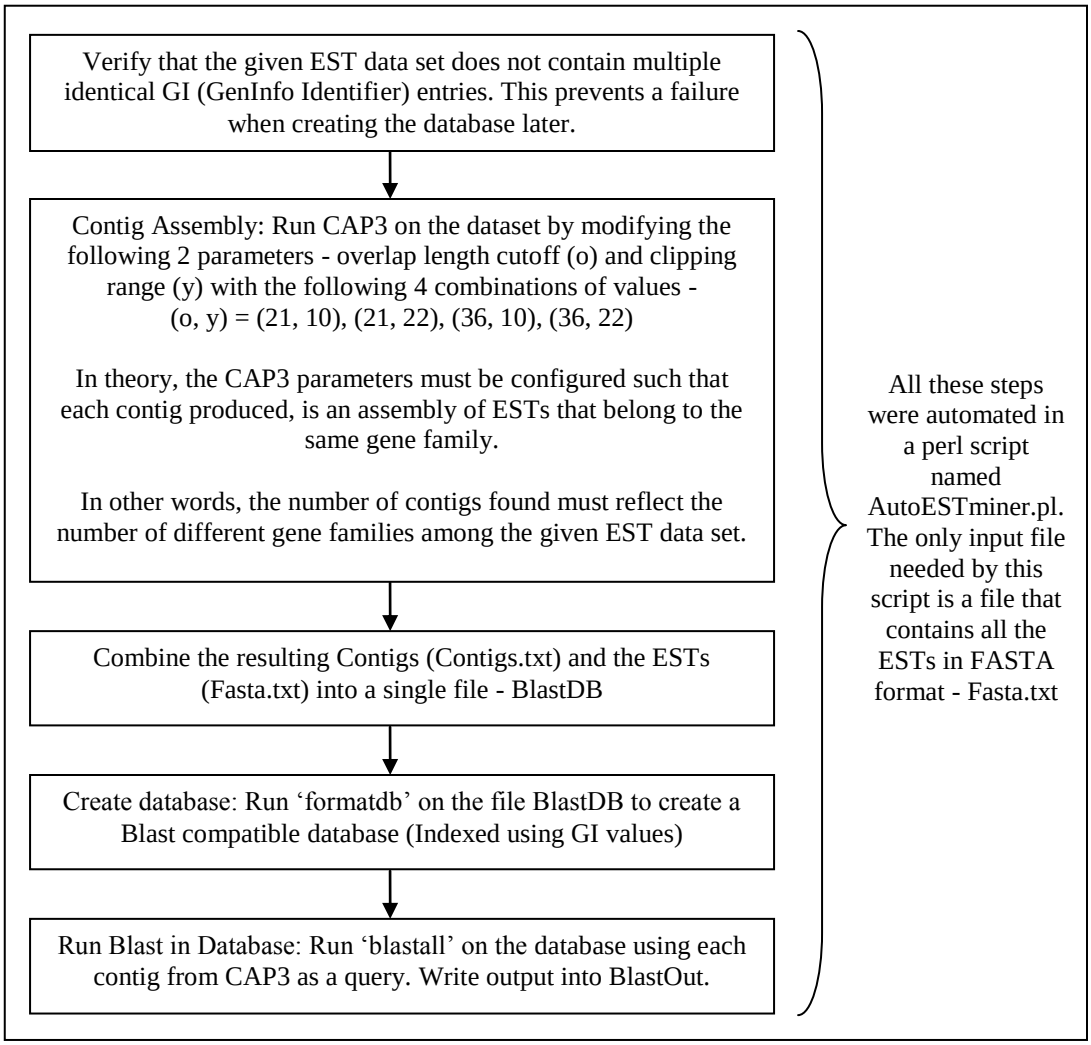

Figure 2.1 - Steps to generate the input used by ESTminer

1_ExtractpHaps.pl - This script requires an input file called *phapin.txt* (aka *snp_est_seqs.txt_haplotypes* by ESTminer) which is generated by ESTminer. This script finds the position of ESTs using information from NCBI MapViewer and uses these positions to try and locate the pHaps on the genome. The start and stop positions of pHap ESTs could be found easily because they are indexed (in MapViewer) by EST accession number. However, locating pHap contigs posed an interesting problem. They were located by first finding the ESTs assembled on either end of that contig. The start position of one of these ESTs and the stop position of the other would thus give us the start and stop positions of the pHap contig itself.

2_MappHaps.pl - This script took the start and stop positions of the pHaps (found by 1_ExtractpHaps.pl), and tried to map them onto the known genes in the At chromosomes (the At genome has 5 chromosomes). For each pHap, the way in which they overlap (or not) with known genes was recorded and tabulated.

3_CountGeneFreq.pl - This script was used to find out the characteristics of the known genes that have been uniquely mapped into (by only one pHap per gene). This is a useful statistic since ESTminer is expected to find only one pHap for every gene.

A fourth script (0_BatchRun.pl) was created to run these 3 analysis scripts one after another for 4 times, each using a distinct set of CAP3 parameters.

2.2. ANALYSIS OF THE TECHNIQUE USING ARABIDOPSIS THALIANA

In their 2004 paper, Nelson et al. used 196K *Glycine max* (Soybean) ESTs to generate pHaps. In this analysis, ESTminer was run on a set of 110K *Arabidopsis thaliana* ESTs (of the Columbia ecotype). The set of 110K Columbia At ESTs were chosen by the following procedure -

1. Retrieved 490,931 Columbia ESTs from GenBank.
2. Discarded ESTs which were not yet mapped into the genome by MapViewer.
3. Of the remaining 346,849 ESTs, selected 110,000 ESTs at random.
4. Note: Only 110,000 were chosen due to the system memory limitations of CAP3.

Since Dr. Nelson stated that the output of CAP3 is critical to results, four different analyses were performed - with two different values for each of the critical parameters - "overlap length cutoff" (-o), and "clipping range" (-y). The four different sets of parameters used, in the form (o, y), were (21, 10), (21, 22), (36, 10), and (36, 22). (o, y) = (21, 10) were the default parameters suggested by Nelson et al. for the Soybean ESTs.

2.2.1. ESTminer results

Table 1 shows a summary that was automatically created by the 1_ExtractpHaps.pl script. Notice how the different values for the "overlap length cutoff"

parameter (-o) did not significantly differ in their results. On the other hand, a small change in the "clipping range" parameter (-y) changed the results quite considerably.

Table 2.1 - Summary of results from ESTminer (generated by 1_ExtractpHaps.pl)

	o21y10	o21y22	o36y10	o36y22
Number of contigs generated by CAP3	12444	12784	12446	12786
Number of CAP3 contigs from which ESTminer produced no pHaps	4012	4275	4012	4275
Total number of pHaps found by ESTminer	16320	16438	16321	16439
Number of pHap contigs constructed	5603	5627	5604	5628
Number of pHap ESTs found	10717	10811	10717	10811
Number of distinct families that contain valid pHaps	8423	8497	8425	8499

2.2.2. Gene mapping results

To analyze the accuracy with which ESTminer's pHaps compare with the known genes in NCBI MapViewer, the script considered the 4 different ways in which a pHap could overlap with a known gene with respect to their start and stop positions. These 4 types are shown in Figure 3.

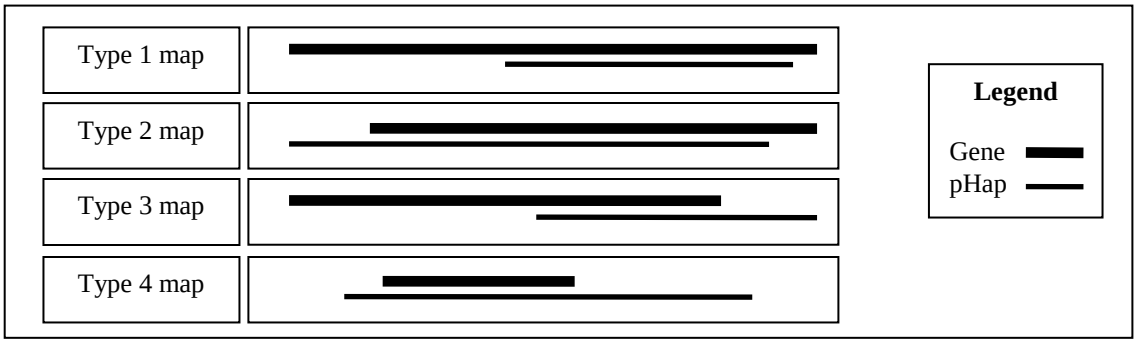

Figure 2.2 - The 4 types of mapping seen between pHaps and genes

A fifth type of mapping would be when a pHap does not overlap with any of the known genes. In this analysis, they are called "unmapped pHaps". They could potentially be At genes that are not identified yet.

Before comparing pHaps with the known genes in NCBI MapViewer, the 1_ExtractpHaps.pl script needed to find which of the 5 chromosomes each pHap belonged to, and their start and stop positions within that chromosome. Some of the ESTs

and genes were listed in two or more different locations on the genome by MapViewer. Because of this, duplicate entries for these pHaps were created - one for each different location in the genome. This is why the number of pHaps used for genome mapping is usually slightly greater than the number of pHaps that ESTminer produced. Figure 4 shows a part of a MapViewer file showing an EST with two different locations.

Start	**Stop**	**Accession number of EST/gene**	
…			
799138	802583	BX815050	(First mapping of this EST in this chromosome)
803035	804300	BP808311	
…			
803040	804290	BX815050	(Same EST mapped in another location in the same chromosome)
…			

Figure 2.3 - Part of a MapViewer file showing two locations for the same entry

Using perl scripts, it was possible to find how often this occurs - 0.004% of ESTs were found to map (in MapViewer) into two or more different locations on the same chromosome. Some of these ESTs are shown in Table 2.

Table 2.2 - ESTs with multiple locations in MapViewer

EST Accession number	Number of locations	Chromosome
AK175799	14	ch4
BP815464	14	ch4
AV824197	5	ch1

Similarly for genes, 0.005% of the genes were found to map (in MapViewer) into two or more different locations on the same chromosome. Some of these genes are shown in Table 3.

Table 2.3 - Genes with multiple locations in MapViewer

Gene ID or name	Number of locations	Chromosome
ATPP2	10	ch1
ATPP2	10	ch2
AT	7	ch3
AT	6	ch4
AT	6	ch5
AT	5	ch1

Since these percentages are very small compared to the total number of pHaps, it can be safely assumed that creating duplicate pHaps does not significantly skew the analysis. Besides, by including these additional start and stop positions for a pHap, the mapping is more complete than if they were ignored.

In the case of pHap contigs, the 1_ExtractpHaps.pl script deduced the start and stop positions from the ESTs that were used to create that contig. However, this caused a rare problem when ESTminer combined ESTs, from distant positions in a chromosome, into the same pHap contig. This resulted in pHap contigs with start and stop positions that were much farther apart than the actual length of the contig sequence. They were easy to spot since they usually overlapped more than 1000 genes (mostly Type 4 maps). With this much in mind, a brief count of pHap mapping was performed by the 2_MappHaps.pl script (Table 4).

Table 2.4 - Number of distinct pHaps that map onto one or more genes

CAP3 Parameters	Type 1	Type 2	Type 3	Type 4	Mapped pHaps	UnMapped pHaps	Total pHaps used
o21y10	14844	598	519	109	16234	116	16350
o21y22	14939	607	521	112	16351	116	16467
o36y10	14845	598	519	109	16235	116	16351
o36y22	14940	607	521	112	16352	116	16468

Note that the above counts are only of *distinct* pHaps. Some pHaps overlapped with more than one gene (with same/different map types). Similarly, a gene could have multiple pHaps map onto it (with same/different map types). If a pHap mapped onto 3 genes with types 1, 1, and 3 respectively, then it is counted once in the ‘Type 1’ column above and once more in the ‘Type 3’ column above. This is why the 4 types in the table above do not add up to the total number of mapped pHaps. Later, all the different mappings are analyzed in detail to classify them as acceptable or unacceptable.

2.2.3. pHap-family distribution

The distribution of pHaps found in the same family was an important result. In all 4 sets of CAP3 parameters, slightly more than 85% of pHaps appeared to be singletons without additional family members. Table 5 shows the number of pHaps obtained per family and the distribution of such families among all the families that produced pHaps. Just as in the experiment by Nelson et al, the distribution shows a large concentration of families with only one pHap each. It was also noted that the distribution is quite similar for all 4 CAP3 parameters used - indicating that changing CAP3 parameters did not make much of a difference. With later results this will become more apparent.

Table 2.5 - pHap-family distribution

Number of pHaps found in the same family	Percentage of such families out of 8423 (o21y10)	Percentage of such families out of 8497 (o21y22)	Percentage of such families out of 8425 (o36y10)	Percentage of such families out of 8499 (o36y22)
1	85.21%	85.23%	85.20%	85.22%
2	4.44%	4.40%	4.44%	4.40%
3	2.81%	2.80%	2.81%	2.80%
4	1.60%	1.65%	1.60%	1.65%
5	1.09%	1.09%	1.09%	1.09%
6	0.81%	0.84%	0.81%	0.84%
7	0.69%	0.66%	0.69%	0.66%
8	0.56%	0.56%	0.56%	0.56%
9	0.40%	0.38%	0.40%	0.38%
10	0.28%	0.28%	0.28%	0.28%
>10	2.10%	2.11%	2.11%	2.12%

2.2.4. Step-by-step filtering of valid/invalid pHaps

The pHaps generated by ESTminer using the CAP3 parameters o = 21 and y = 10, were carefully classified into different categories and analyzed as described below.

1. 16350 distinct pHaps were obtained from ESTminer using the o21y10 parameters.

2. All the pHaps that mapped onto only one gene each or no gene at all (15189 pHaps) were kept aside and analyzed later in Step 4 where pHaps were validated according to how many of them map into the same gene.

3. The remaining 1161 pHaps include only those that map into two or more genes. They were classified according to how many genes they each map into.
   a. All the pHaps that mapped onto more than 10 genes were considered invalid. Only 7 such pHaps were found. For example, Contig7194:1 (mapped 6067 genes), Contig6105:1 (mapped 5278 genes), etc. This happened when ESTminer combined ESTs, from distant positions in a chromosome, into the same pHap contig such that its start and stop positions had thousands of genes in-between.
   b. The pHaps that map onto 2 genes each (1081 such pHaps) were analyzed. These pHaps were classified according to the sequence with which they map into genes. With this classification, the three most common ways in which pHaps mapped into two genes were found. These are shown in Table 6.

Table 2.6 - The 3 most common ways in which a pHap mapped onto 2 genes

Map sequence	Freq	Sample pHaps
Type 1 Type 1	448	AA395556, Contig31:1
Type 3 Type 1	263	AU238629, Contig1:1
Type 1 Type 2	255	AV797203, Contig5:1

Figure 5 shows some sample map sequences for the types mentioned.

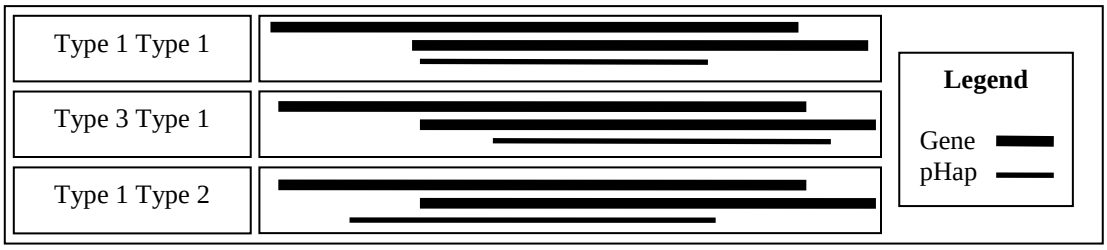

Figure 2.4 - The 3 most common ways in which a pHap mapped onto 2 genes

It was concluded that these unusual maps were mostly because of gene overlaps in MapViewer. Gene overlaps occur when potential (but unconfirmed) genes are positioned over each other. Since this was a downside of MapViewer, it was decided to leave these 1081 pHaps as inconclusive towards the performance of ESTminer.

c. The pHaps that map onto 3 genes each (28 such pHaps) were analyzed.

Table 2.7 - How three pHaps map into the same gene

Map sequence	Freq	Sample pHaps
Type 3 Type 4 Type 2	12	AV518488, Contig1945:1
Type 4 Type 4 Type 2	6	AV793704, Contig11560:1
Type 3 Type 4 Type 4	4	AV531450, Contig6811:1

As can be seen from Table 7, most of these pHaps mapped with a Type 4 map in-between. This implied that the maps were mostly because of the rare ESTminer problem explained in Step 3a and hence these pHaps were invalid.

d. The pHaps that mapped onto more than 3 genes each (48 such pHaps) and less than 10 were analyzed. Here too, most of the pHaps mapped onto multiple genes with Type 4 maps. So they were considered invalid. However, a small minority of these pHaps were found to map into multiple genes in very unique ways. These were all because of unusual positioning of genes within MapViewer. Though quite interesting, these odd mappings were ignored because they occurred too rarely to affect the overall analysis.

4. The 15189 pHaps that mapped onto only one gene each or no gene at all were analyzed.
   a. 116 of these pHaps mapped into areas with no known genes (unmapped pHaps) were ignored. Whether valid or invalid, ‘116’ was too few to be significant to either cause. These pHaps point to previously unmapped locations and could potentially point to previously unidentified genes.
   b. The remaining 15073 pHaps were analyzed. This analysis is shown in Table 8.

Table 2.8 - Analysis of pHaps that map onto one gene each

Valid number of pHaps that map into the same gene	Number of such genes	Total number of such pHaps	Percentage of pHaps (out of total 15073 pHaps analyzed)
1	4951	4951	32.85%
2	909	1818	12.06%
3	250	750	4.98%
4	151	604	4.01%
5	92	460	3.05%
6	84	504	3.34%
7	56	392	2.60%
8	42	336	2.23%
9	30	270	1.79%
10	29	290	1.92%
11	17	187	1.24%
12	12	144	0.96%
13	16	208	1.38%
14	14	196	1.30%
15	10	150	1.00%
16	11	176	1.17%
17	5	85	0.56%
18	8	144	0.96%
19	5	95	0.63%
20	3	60	0.40%
57	1	57	0.38%
61	1	61	0.40%
62	1	62	0.41%
63	2	126	0.84%
71	1	71	0.47%
74	1	74	0.49%
97	1	97	0.64%
106	1	106	0.70%
119	1	119	0.79%
120	2	240	1.59%
126	1	126	0.84%
130	1	130	0.86%
144	1	144	0.96%
150	1	150	1.00%
394	1	394	2.61%

Table 8 shows that only 4951 pHaps out of 15073 pHaps (32.85%) mapped into a gene which has only one pHap map into it i.e. which have a one-to-one correspondence between pHap and gene.

2.3. CONCLUSION OF ANALYSIS

The analysis of the final 15073 pHaps was performed by the 3_CountGeneFreq.pl script that counted the number of pHaps that map into the same gene, and then distributed the genes according to that number. This analysis (Table 8) showed us that – out of the 16350 pHaps produced by ESTminer, only 30.28% uniquely identified a gene in *Arabidopsis thaliana*. This suggested that ESTminer was not working as the authors intended.

3. GI USING NEGATIVE SELECTION PATTERNS

This chapter introduces a method of gene identification proposed by Dr. Ronald Frank in 2006 (Frank et al., 2006). The technique made use of the massive NCBI databases and their online local alignment search tool - BLAST. Since all the steps of this technique could be performed on a computer (with access to NCBI online services), it only seemed practical to try to automate as much of the technique as possible. This chapter describes the originally published automation of the technique and the subsequent improvements made since. But first, the rationale behind using negative selection patterns for gene family identification needs to be explored.

3.1. AN INTRODUCTION TO NEGATIVE SELECTION

The most popular and well understood mechanism of evolutionary adaptation is *natural selection*. It is the process by which genes favorable to an organism in its environment are carried over to future generations whereas deleterious genes are not. This means that over time, future generations of the organism will be better adapted to their environment. A common mechanism for such adaptation is *negative selection*. If an organism has genes that are deleterious to its survival, it subsequently loses its chance to reproduce, and its deleterious genetic information is lost. Over time, future generations of the organism are more likely to contain the genes of more successful survivors and reproducers.

In today's understanding of evolutionary mechanisms, gene duplication is widely considered to play a major role (Taylor et al., 2004). A duplication event can cause any region of DNA to be duplicated - a region that contains one or more genes, a whole chromosome, or sometimes even the entire genome. Copies of the same gene that exist due to a gene duplication event are called paralogs. At least one of the paralogs will retain the original function as long as it is beneficial to the organism. Because of this "backup copy" the other paralogs are free from selective pressure and thus accumulate more mutations into future generations. This may lead to altered function (subfunctionalization), a new function (neofunctionalization), or loss of function. As an example for subfunctionalization, consider the duplication of a protein-coding gene that

encodes a protein for the root of a plant. After several tens of thousands of years, a paralog of the gene might still encode the original protein. But instead of being expressed in the plant's roots, mutations in the regulatory sequence (that controls the expression of the gene) may cause it to be expressed in the plant's stem. More often than not, mutations create new proteins which are deleterious to the plant, causing death (or failure to reproduce), and therefore not passed on to the next generation (negative selection). However, on the rare chance that the new protein turns out to be beneficial for the plant, then it is called a neofunctionalization.

If mutations occur evenly across a gene, and negative selection allows only certain mutations to be carried over into future generations, then a deterministic pattern of differences between paralogs can be seen. In particular, for protein-coding genes, synonymous substitutions result in the same protein and are thus carried over into future generations. Non-synonymous substitutions result in a new protein which is either deleterious to the organism (common) or beneficial to the organism (rare). Hence, coding regions (the region that encodes the protein) of paralogs that have subfunctionalized via changes in regulatory elements should exhibit more synonymous substitutions than non-synonymous ones. This mechanism appears to be very common in plants, causing a large proportion of plant genes to belong to gene families (Lockton et al., 2005). If this is the case, then most plant gene families can be identified by a pattern of bias toward synonymous substitutions between contigs assembled from related ESTs.

3.2. A NEGATIVE SELECTION PATTERN

In Dr. Frank's GI technique, the number of base differences between potential paralog pairs is counted with respect to their positions in a codon. The rationale behind the NSP (negative selection pattern) based technique is explained as follows. Table 9 shows all the 20 amino acids and the corresponding codons that encode them. Of the 64 codons, 3 of these do not encode amino acids (UAA, UAG, and UGA). They are instead used in DNA as *stop codons* - tags which mark the end of a coding region.

Table 3.1 - Amino acids encoded by various triplets of nucleotides (codons)

	Second Position Differences								
First Position Differences	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	Third Position Differences
	UUC		UCC		UAC		UGC		
	UUA	Leu	UCA		UAA	Stop	UGA	Stop	
	UUG		UCG		UAG	Stop	UGG	Trp	
	CUU		CCU	Pro	CAU	His	CGU	Arg	
	CUC		CCC		CAC		CGC		
	CUA		CCA		CAA	Gln	CGA		
	CUG		CCG		CAG		CGG		
	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	
	AUC		ACC		AAC		AGC		
	AUA		ACA		AAA	Lys	AGA	Arg	
	AUG	Met	ACG		AAG		AGG		
	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	
	GUC		GCC		GAC		GGC		
	GUA		GCA		GAA	Glu	GGA		
	GUG		GCG		GAG		GGG		

From Table 9 it is easy to see that a change in the third position of a codon is most likely to be synonymous. For example - CCU, CCC, CCA, and CCG all encode the same amino acid. This is known as a 4-fold redundancy. A change in the first position of UUA to CUA does not cause a change in the encoded amino acid. This is called a 2-fold redundancy. Table 10 shows all such first position redundancies. Similarly, a 3-fold redundancy exists between the codons - AUU, AUC, and AUA. Note how any change in the second position of a codon causes the encoded amino acid to change. Hence, any second position substitution is always non-synonymous.

Table 3.2 - First position 2-fold redundancies

UUA	Leu	UUG	Leu	CGA	Arg	CGG	Arg
CUA		CUG		AGA		AGG	

Of the 61 different codons that produce 20 different amino acids, 8 codons are 2-fold redundant (Table 10) in the first position, there are no redundancies in the second position, while in the third position, 24 codons are 2-fold redundant, 3 are 3-fold redundant, and 32 are 4-fold redundant. The distribution of differences between two subfunctionalized paralogs at the first, second, and third positions of each codon, show that the third position differences occur much more frequently than first or second position differences. Dr. Frank defined a discernible threshold for this NSP as follows -

"If differences appear non-random with respect to their position in a codon, and third position differences are more than 3 times the first position differences, and all differences are distributed as to satisfy the relationship $3^{rd} > 1^{st} > 2^{nd}$, then we can conclude that the contigs represent different genes. However, if these criteria are not met, we do not conclude that the contigs necessarily represent the same gene."
(Frank et al., 2006).

It must be emphasized that such a technique can only identify gene families with protein-coding members that have diverged after a gene duplication event, and thus show a typical negative selection pattern. Since this specific evolutionary mechanism is common in plants (Lockton et al., 2005), Dr. Frank's NSP technique is better suited to identify gene families in plant genomes.

3.3. INITIAL IMPLEMENTATION OF THE NSP-BASED GI TECHNIQUE

The first attempt at automating this GI technique was published by Frank et al. in 2006 (Frank et al., 2006). The automation used PERL scripts which were designed to run on UNIX, Linux, or Windows platforms. It either took a set of related ESTs as input, or a query sequence which could then be used to find the related ESTs. The query must be a protein-coding sequence which is believed to be one of many paralogs (belonging to a gene family). The query could also be an orthologous sequence (sequences with similar function but from different genomes) from a related species. The query sequence is submitted to the online NCBI BLAST service to search for ESTs that are similar to it and which belong to the organism in question. The ESTs returned by the BLAST search were assembled into contigs using an application called *AssemblyLIGN*. The open reading frames (the part of the sequence than encodes the protein) of the resulting contigs are identified and recorded using another application called *MacVector*. PERL scripts are then used to submit each pair of contigs to a pair-wise sequence alignment algorithm called *bl2seq* (BLAST for 2 Sequences) which is also an online NCBI service. The script then counts the number of 1^{st}, 2^{nd} and 3^{rd} position differences between each contig pair and tabulates the results as a matrix stored in a file. Figure 6 shows the steps involved in this automation.

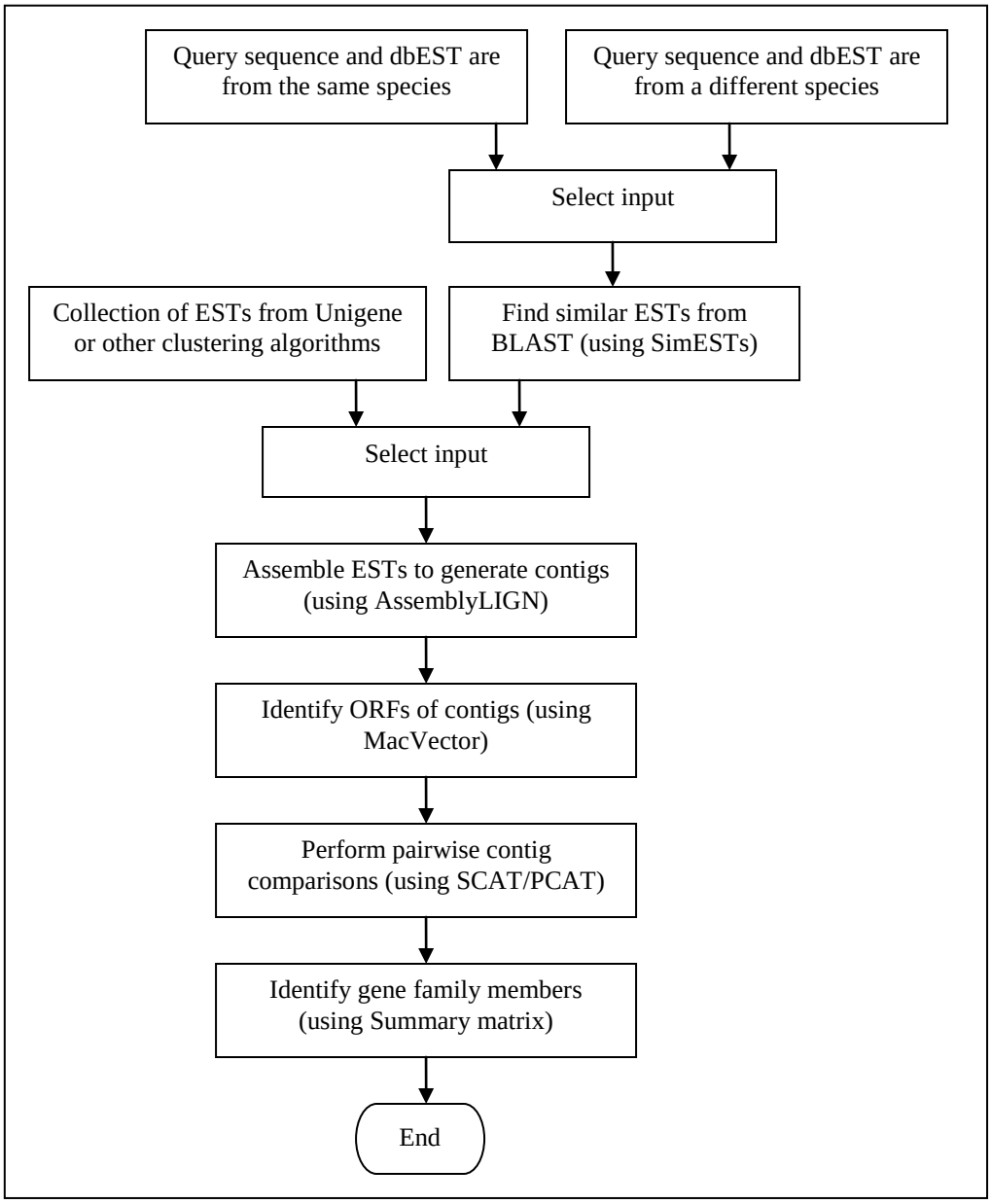

Figure 3.1 - Steps in automation of the NSP-based gene family identification technique

3.3.1. Automation

The steps that are automated in the above implementation are called *SimEST* and *SCAT*. Other steps like the BLAST search, contig assembly, and ORF identification have to be performed manually using external applications. The BLAST search, in particular, is an online NCBI service which can be accessed in a browser. The search results have to be manually saved in a text file. The automated SimEST script reads the EST accession

numbers from this file and retrieves the actual sequences using another NCBI web service known as e-Utils (Entrez Programming Utilities). Once the sequences are obtained and saved in a file, they have to be assembled into contigs by manually submitting them into the AssemblyLIGN application. The ORFs of the resulting contigs are then identified by analyzing them in the MacVector application. The contigs and their ORFs have to be stored in files in a predetermined format so that the SCAT script can submit each pair of contigs to the *PCAT* script. PCAT aligns two given contigs using bl2seq and then counts the 1st, 2nd and 3rd position differences between them with respect to the ORFs of the contigs. After PCAT is run on every contig pair, SCAT creates a matrix containing 1st, 2nd and 3rd position differences between every contig pair.

3.3.2. Validation

Frank et al. validated this implementation using the well known PAL gene family from *Arabidopsis thaliana*. The PAL family has 4 known members - PAL1, PAL2, PAL3, and PAL4. By submitting PAL1 as the query, the remaining paralogs were expected to be found. After going through all the steps, SCAT created Table 11.

Each cell in the table indicates the number of 1st, 2nd and 3rd position differences separated by commas. An 'NS' indicates that no significant similarity was found between the contigs. A 'NO' indicates that bl2seq returned an alignment of the contigs, but the ORFs identified on those contigs did not overlap.

Table 3.3 - Distribution of 1st, 2nd, and 3rd position differences between contigs

	Contig3	Contig4	Contig5	Contig6	Contig7	Contig8	Contig9
Contig1	20, 6, 33	14, 4, 43	NS	NS	NS	NS	18, 5, 32
Contig3	***	45, 19, 146	NS	19, 7, 70	NS	15, 7, 55	5, 5, 6
Contig4	***	***	NS	3, 4, 4	NS	1, 2, 1	10, 4, 27
Contig5	***	***	***	NO	NS	NS	NS
Contig6	***	***	***	***	6, 5, 39	4, 3, 7	NS
Contig7	***	***	***	***	***	NS	NS
Contig8	***	***	***	***	***	***	NS

(Frank et al., 2006)

By applying the threshold for the NSP as defined by Frank et al. (see section 3.2), Table 11 shows that Contig1 represents a different gene from Contig4. The distribution of differences (1st position: 14, 2nd position: 4, 3rd position: 43) definitely shows a non-

random distribution of differences, the third position differences are 3.1 times that of first position, and overall $3^{rd} > 1^{st} > 2^{nd}$. Similarly, Contig3 and Contig4 also represent potential paralogs. Also note how Contig6 and Contig8 have very few differences with Contig4. They are most likely the same gene, but with a few sequencing errors. With this much information, 3 distinct PAL paralogs have been clearly identified, each represented by Contig1, Contig3, and Contig4.

Table 3.4 - Percent similarity of potential paralogs (contigs) with known PAL genes

	GeneA	GeneB	Gene C		
	Contig1	Contig3	Contig4	Contig6	Contig8
PAL1	96%	76%	86%	83%	86%
PAL2	79%	76%	100%	97%	98%
PAL3	81%	83%	76%	77%	77%
PAL4	73%	99%	76%	85%	86%

(Frank et al., 2006)

In Table 12, the percent similarity of the potential paralogs with the 4 known PAL genes is calculated using bl2seq alignments. Contig1 is most similar to the PAL1 gene (which was the protein-coding query gene). Contig3 represents the PAL4 gene and Contig4, Contig6, and Contig8 all appear to represent the PAL2 gene. Clearly, this experiment provided limited validation for Frank et al. to conclude that the NSP-based GI technique worked as expected. It was able to correctly identify some contigs as potential paralogs to the query sequence. Furthermore, by increasing the size of the set of ESTs assembled into contigs, the chance of identifying new potential paralogs can be increased.

3.3.3. Issues

One problem with this implementation is that it is not fully automated. It requires several instances of manual intervention to run the three external applications - BLAST, AssemblyLIGN, and MacVector. AssemblyLIGN and MacVector, being GUI (Graphical User Interface) applications, could not be automatically controlled by a PERL script. Also, they are Macintosh-based applications whereas SimEST and SCAT are only supported on UNIX, Linux, and Windows platforms. This required the user to switch from one platform to another in-between steps during the process.

The alignment between contigs was performed by *bl2seq* which is a local alignment program. This means that the most similar subsets of each sequence are aligned against each other. What is ideally desired is that two contigs must be aligned for the entire length of their ORFs. This is known as *global alignment*. Figure 7 shows how these two types of alignment might align the same two sequences.

Global Alignment

```
Sequence 1: AGACTGAGAG-GTGACCTGACCGT
Sequence 2: A-CTG-AG-GAG-G---TGACC-T
```

It forces an alignment over the full length of both sequences.

Local Alignment

```
Sequence 1: AGACTGAG-AGGTGACCTGACCGT
Sequence 2: --ACTGAGGAGGTGACCT------
```

It determines similar regions between sequences by comparing sub-sequences of all possible lengths to find the optimum alignment.

Figure 3.2 - The two methods of sequence alignment

Incorrectly inserted gaps can shift the sequence such that the position of nucleotides within their codons is incorrect. This problem is more likely to occur in local alignments rather than global alignments. However, it is possible to more closely analyze each inserted gap with respect to their surrounding nucleotides or a nearby stop codon, to determine whether the gap has been inserted correctly or not. The user can perform this manually using MacVector. However, gaps are generally uncommon in this implementation since the aligned contigs are very similar to each other and their ORFs are usually of equal length. This is why the validations ran successfully despite using the local alignment algorithm bl2seq.

3.4. A FULLY AUTOMATED IMPLEMENTATION

To fully automate the NSP-based GI technique, it was first necessary to find non-GUI alternatives to AssemblyLIGN and MacVector. PERL scripts are capable of automating only local command-line applications and online browser-based services. CAP3, a command-line contig assembly program, was chosen as an alternative to AssemblyLIGN. The NCBI ORF Finder, a browser-based online application was chosen as an alternative to MacVector. NCBI BLAST search was available as an online web-

service and could be automated in PERL too. Since all portions of the original implementation now had alternatives that could be controlled by PERL scripts, it was possible to create a continuous streamlined automation that ran from start (input of query protein) to stop (tabulated comparison of potential paralogs). The following section describes the implementation of this automation and validates its operation using PAL genes from *Arabidopsis thaliana*.

3.4.1. CAP3 (Contig Assembly Program, 3rd Generation)

Huang et al. designed CAP3 to assemble short reads (ESTs) into long sequences (contigs). Algorithms in CAP3 identify and compute overlaps between reads, construct multiple sequence alignments of these reads, and generate consensus sequences (Huang et al., 1999). In other words, given a set of ESTs and specific parameters of operation, CAP3 attempts to combine as many of these ESTs together to form the smallest possible set of long contigs. ESTs that could not be combined with other ESTs to produce a contig are known as *singletons*. Below, the two parameters of CAP3 that are relevant to the GI technique are described:

Overlap length cutoff (o) - This is the minimum number of bases on two ESTs that need to overlap before being considered for inclusion into the same contig. In the earlier implementation, AssemblyLIGN set its equivalent parameter to 20 bases. In CAP3, this parameter cannot be lower than 21. Hence, a default value of 21 bases was used for *overlap length cutoff (o)*.

Clipping range (y) - This is the number of bases that will be clipped (discarded) on both ends of an EST. The process of generating ESTs by reading DNA sequences (shotgun sequencing) causes it to be less accurate near the ends of the EST and more accurate near the middle. Clipping bases from the ends improves the overall accuracy, at the cost of losing information. In the 2006 implementation, AssemblyLIGN did not clip its ESTs ($y = 0$). However, CAP3 requires a minimum of 6 bases to be clipped ($y > 5$). It is noted that the results of CAP3 is heavily influenced by changes in y. Through multiple trials, it was decided that 50 bases was a reasonable value for y and it produced the best results.

Overlap percentage identity cutoff (p) - This is the percentage of bases that need to perfectly match each other in the overlap region between two ESTs. AssemblyLIGN used 100% in its equivalent parameter. CAP3, being more rigid in its alignments, could not create contigs for p > 95%. Between 90% and 95%, CAP3 generated too many contigs because of the fewer overlaps found. Through multiple trials, p = 90% is selected as a reasonable value for this parameter.

Despite hard-coding default values for each parameter into the script, it did allow the user to specify custom values for these 3 CAP3 parameters. In particular, it was noted that changes in clipping range (y) significantly affected the resulting contigs. This is most likely due to the variable accuracy levels among different sets of ESTs.

3.4.2. NCBI ORF Finder (Open Reading Frame Finder)

The ORF Finder is a browser-based online analysis tool which finds all open reading frames of a selectable minimum size in a given sequence. An Open Reading Frame (ORF) is the part of a nucleotide sequence that encodes a protein. This tool identifies all of the possible ORFs of a given sequence using the standard genetic code for the start and stop codons. This code states that the start codon is usually AUG. This is the most common codon that marks the start of an ORF. In rare cases, an ORF may start with a codon other than AUG, like GUG, CUG, or UUG. The stop codons always mark the end of an ORF. These are always - UAA, UAG, or UGA.

The ORF Finder takes the input contig and finds the start and stop codons on it. If several such codons are found, then there will be more than one possible ORF on the contig. In this implementation, the longest ORF found on each contig was selected.

3.4.3. Automation

The operation of the script is summarized as a flowchart in Figure 8. The PERL script that fully automates this implementation is named *Auto.pl*. It is designed to work on all platforms supporting a PERL environment including Macintosh-based OS X. The input to this script is the protein sequence whose potential paralogs need to be found. Optionally, the user can input customized values for CAP3 parameters overlap cutoff (o), clipping region (y), and overlap percentage identity cutoff (p).

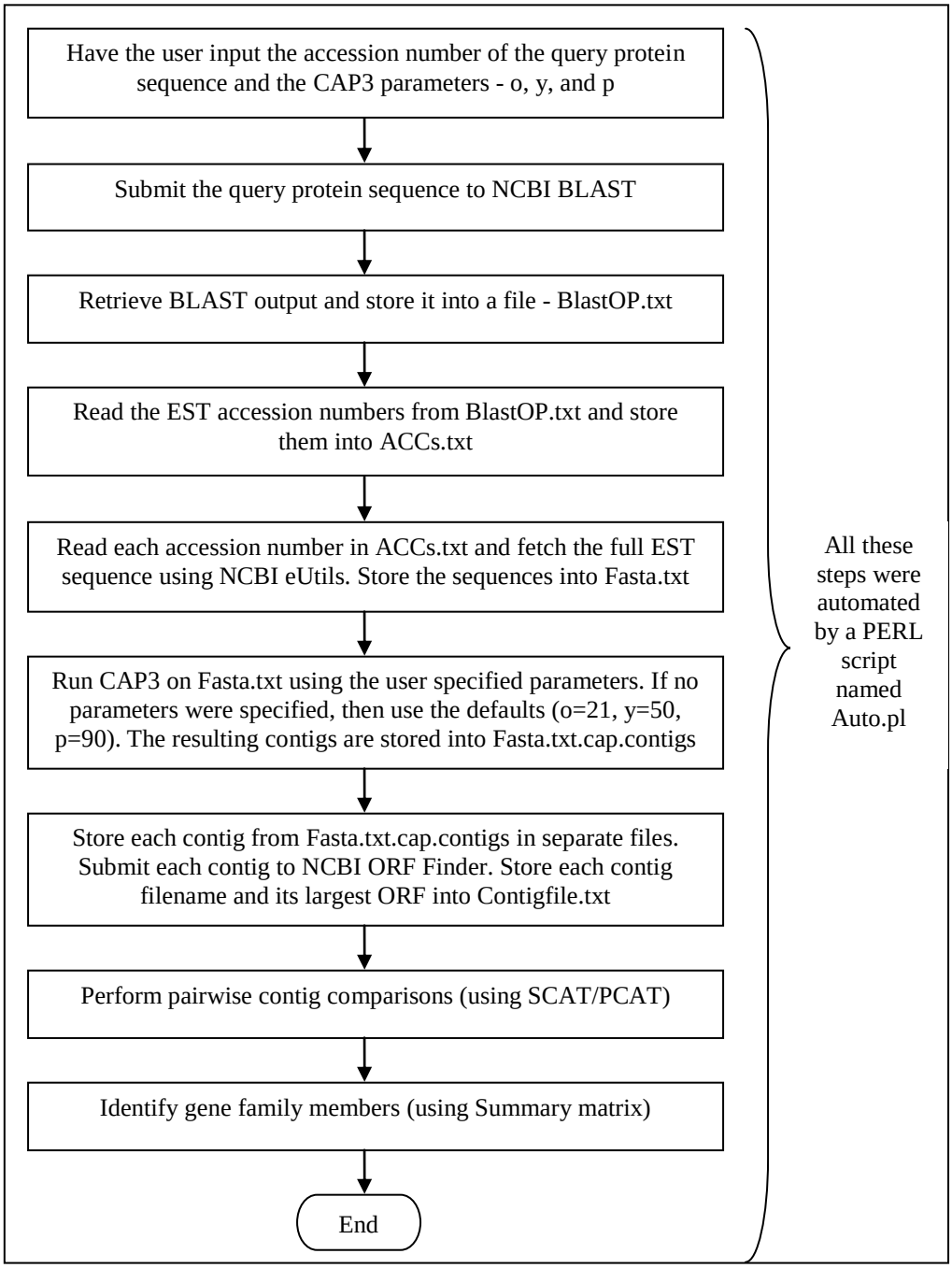

Figure 3.3 - Fully automated steps in NSP-based gene family identification techn

Initializations

- Get the accession number of the query protein sequence as input from the user
- Get the following CAP3 parameters as input from the user. If the user skips this step, then use the following default parameter values

- o, overlap length cutoff (default value = 21)
- y, clipping range (default value = 50)
- p, overlap percent identity cutoff (default value = 90)

BLAST Search

- Submit the input query sequence to the BLAST program and wait until the query is completed. Do the following during the wait period:
- Extract the BLAST Request ID (RID) from the returned webpage
- Repeat the following every 2 seconds (Poll for results)
  - Read the website that contains the status of the Request ID
  - Extract status information (waiting/failed/unknown/ready/nohitsfound)
  - Indicate the current status to the user - Print a dot (.) if current status is 'waiting'
  - If the results are ready to be fetched (status = 'ready'), then exit this loop
- Fetch the BLAST results and store them into a file called BlastOP.txt

Extracting Accession Numbers and fetching their EST sequences

- For each line of BlastOP.txt that contains an accession number, repeat the following
  - Extract the acc no. in the line and store it into ACCs.txt
  - Submit this acc no. as a request to Entrez Utilities' efetch.fcgi service
  - Retrieve the sequence returned and append it into a file - Fasta.txt

CAP3 Automation

- Run the CAP3 application on the command-line. Pass the path to Fasta.txt and the three parameters o, y, and p to CAP3 as command-line arguments. If the user did not specify these parameters, then run with the default values as shown below -

OS X/UNIX/Linux	`>./cap3 Fasta.txt -o 21 -y 50 -p 90`
Win32	`>cap3.exe Fasta.txt -o 21 -y 50 -p 90`

NCBI ORF Finder Automation

Of the 8 output files generated by the CAP3 program, load the file Fasta.txt.cap.contigs, and repeat the following steps for each contig sequence in this file:

- Submit the sequence to the NCBI ORF Finder, and read the resulting webpage
- Remove the HTML tags to reveal information on all of the ORFs found
- Find the start, stop, and frame of the longest ORF

- Store the full contig sequence into a file named - ContigXY.txt - where X is either '+' or '-' (a '-' indicates that the ORF is in the reverse complement of the sequence), and Y is the contig number
- Append the contig filename and the corresponding ORF start and stop positions into Contigfile.txt

SCAT/PCAT Automation

- Submit Contigfile.txt to the SCAT script so that it generates a tabulated summary of the pairwise contig comparisons

3.4.4. Validation

This automation is validated using the known PAL genes in *Arabidopsis thaliana*. This would allow us to compare its performance against the results obtained through the semi-automated technique used previously by Frank et al. and ESTminer. The accession number for the PAL1 gene (AAK76593) is used as input to Auto.pl and the CAP3 parameters are set to the default values. The script was run on the command line as -

OS X/UNIX/Linux `>perl Auto.pl AAK76593`

Win32 `>Auto.pl AAK76593`

When the script finally terminates, a tabulated summary of the pairwise comparisons between contigs is generated and it is stored into a file. The table which is generated as a result of the PAL1 query is shown in Table 13.

Table 3.5 - Distribution of 1st, 2nd, and 3rd position differences between contigs

	Contig1	Contig2	Contig3	Contig4	Contig5	Contig6
Contig1	***	8, 2, 8	0, 0, 2	NS	0, 0, 0	NS
Contig2	***	***	1, 0, 0	NO	12, 12, 16	NO
Contig3	***	***	***	NS	6, 7, 10	NS
Contig4	***	***	***	***	16, 8, 99	10, 6, 35
Contig5	***	***	***	***	***	4, 12, 28
Contig6	***	***	***	***	***	***

It can be seen that Contigs 1, 2, and 3 are very similar to each other and do not appear to show an NSP with any other contig. Contigs 4, 5, and 6 show a clearly non-random pattern of differences between each other. In each, the number of 3rd position differences between pairs is more than 3 times the number of 1st position differences. The

differences in all three pairwise comparisons (except for Contig5 and Contig6) appear to satisfy the rule $3^{rd} > 1^{st} > 2^{nd}$. Since Contig5 and Contig6 do not pass the discernible threshold defined by Frank et al., they do not qualify as potential paralogs. However, Table 14 clearly shows that Contig5 and Contig6 are very similar to the PAL1 and PAL4 genes respectively. Also, Contig4 was found to be most similar to the PAL2 gene.

Table 3.6 - Percent similarity of potential paralogs (contigs) with known PAL genes

	No significant patterns found			Gene A	Gene B	Gene C
	Contig 1	Contig 2	Contig 3	Contig 4	Contig 5	Contig 6
PAL1	97%	84%	77%	86%	99%	76%
PAL2	80%	98%	81%	99%	84%	76%
PAL3	NS	76%	76%	75%	75%	82%
PAL4	NS	76%	97%	75%	75%	99%

3.4.5. Discussion

Three of the contigs generated by CAP3 closely resemble 3 paralogs in the PAL gene family. The distribution of 1^{st}, 2^{nd}, and 3^{rd} position differences between these contigs (Contigs 4, 5, and 6) appear to be non-random and the number of third position differences is at least three times the first position differences. This satisfies two of the three criteria for NSP, as stated by Frank et al. The third criterion, which requires the second position differences to be lesser than the first position differences, was satisfied by two of the three contig pairs. The distribution of differences between the other two Contigs 5 and 6 was calculated as 4, 12, 28. Even though this was clearly a non-random distribution of differences, the number of second position differences was quite larger than the first position differences. One explanation for this is that since ESTs were allowed to overlap with 90% match (CAP3's overlap percent identity cutoff, p = 90), ESTs carrying errors were allowed to assemble into the contigs. It is possible to reduce such errors by clipping EST edges, but if too much of the EST is clipped out, the resulting contigs became shorter in length. This leaves bl2seq with shorter sequences to align and compare. Longer contigs generate a more distinctive NSP than shorter ones. In other words, y must be large enough to remove EST errors, and short enough to allow the creation of longer contigs. This causes the automation to be very sensitive to the clipping range parameter 'y'. The automation must be repeated with different combinations of

CAP3 parameters (particularly y) until distinctive negative selection patterns are seen in the resulting distribution.

3.5. RESULTS AND DISCUSSION

The results of the new automation in Table 13, show less distinctive patterns of negative selection than the older implementation in Table 11. However, comparing Table 14 with Table 12, it can be seen that the same 3 paralogs were represented by three contigs in both implementations. This means that CAP3 correctly generated contigs that were subfunctionalized paralogs, but there were too many errors between them that the negative selection patterns were not distinctive enough. The contig assembly application used in the earlier implementation (AssemblyLIGN) minimizes such errors by restricting overlaps between ESTs to a 100% match. Since AssemblyLIGN is manually operated by a user, errors in the assembly can be intuitively identified and immediately fixed. Errors in ORF identification can be intuitively fixed by the user with MacVector. The trade-off, of course, is the required manual control of the application. In comparison, the CAP3 based automation of the GI technique is quick and efficient. The user can conveniently repeat the automation with different CAP3 parameters until a distinctive negative selection pattern is found.

In conclusion, the newer implementation appears to get the job done quickly and correctly while slightly compromising accuracy. However, this particular negative selection pattern is easy to spot even when there is a high rate of errors in the ESTs. The 2006 implementation by Frank et al. requires occasional manual intervention and hence allows the user more control over each step of the process at the expense of efficiency.

4. CONCLUSION

A fully automated gene family identification program was created which tapped into the information stored on huge online databases that are constantly updated with newly sequenced genomic data. A user simply needed to input the accession number to a protein-coding gene and the script would generate a table of its potential paralogs. The user can then easily identify these paralogs in the organism's genome.

The current implementation of NSP-based GI runs quickly and efficiently except for the BLAST search, where the wait-times on the NCBI server are non-deterministic. Nevertheless, it is only necessary to run BLAST once for a given protein-coding query gene. After BLAST fetches the ESTs that are homologous with the query sequence, the user can run the remaining steps any number of times without repeating BLAST every time. This makes it much faster for the user to try out different CAP3 parameters and find an optimum configuration that produces the negative selection pattern.

In the analysis performed on this implementation, it was found to correctly identify 3 members of the *Arabidopsis thaliana* PAL gene family with moderate accuracy. The biggest drawback of the automation is in its dependence of CAP3. Even though the contigs that it assembled were prone to errors (because of erroneous ESTs), the GI technique was still able to identify distinctive NSP patterns. However, it would go a long way to implement error correction techniques or to generate files containing quality information about ESTs – which can be used by CAP3 to generate more accurate contigs.

The analysis of ESTminer revealed that it did not work as expected. Most of the pHaps identified did not uniquely map into genes. This meant that ESTminer was not correctly identifying potential haplotype sequences i.e. a set of sequences closely linked by DNA polymorphisms. The application depended too heavily on accurate contig assembly and relied on the assumption that each contig generated by CAP3 would be representative of a gene family.

4.1. COMPARISON OF GI TECHNIQUES

A summarized comparison of the two GI techniques is shown in Table 15. It also compares the two implementations of the NSP-based GI technique - the earlier implementation (which used AssemblyLIGN and MacVector), and the newer implementation (using CAP3 and NCBI ORF Finder).

Table 4.1 - Summarized comparison of GI techniques

ESTminer	**NSP using AssemblyLIGN and MacVector**	**NSP using CAP3 and NCBI ORF Finder**
Attempts to find paralogs Genome-wide	Finds paralogs only within one Gene-family at a time	Finds paralogs only within one Gene-family at a time
Based on single nucleotide polymorphisms found between potential paralogs	Based on a negative selection pattern found between potential paralogs	Based on a negative selection patterns found between potential paralog
Uses CAP3 to cluster gene families together throughout the genome	Uses NCBI BLAST to find ESTs which are likely to contain paralogs of the query (in the same gene family)	Uses NCBI BLAST to find ESTs which are likely to contain paralogs of the query (in the same gene family)
Platform independent (Needs a PERL interpreter installed)	Certain parts run on Mac OS X; Other parts run on Win32/Linux (Needs a PERL interpreter installed)	Platform independent (Needs a PERL interpreter installed)
After manually running CAP3 and creating a local BLAST database, it is automated in three scripts: *AlleleFinderV4.1.3* *HaplotypeSorterV1.1* *SNPfindV14.0.8*	AssemblyLIGN and MacVector need to be manually operated between using two scripts: *SimEST* *SCAT*	Fully Automated in a single script: *AutoNSP*
Moderately affected by EST sequencing errors	EST sequencing errors can be manually fixed using AssemblyLIGN or MacVector	Moderately affected by EST sequencing errors

4.2. FUTURE WORK

As future work, the NSP-based GI technique can be extended by querying the automation with orthologs from related organisms rather than with paralogs in the same organism. Even with the current implementation, a protein sequence from some organism can be submitted, while using BLAST to search for ESTs from another organism. This kind of operation needs to be further tested. Another possible modification is to submit the paralogs found by the script back to the script. This way a tree can be automatically generated that shows the entire phylogeny of genes in a species that evolved after gene duplication events.

Furthermore, it would be useful to find an alternative to AssemblyLIGN that can be automated into a perl script. For the same reason, it would also be useful to find an alternative of CAP3 which gives the user more freedom in manipulating the contigs. CAP3 was originally designed to assemble contigs for entire genomes. If the source code for CAP3 can be obtained from its authors, then it might be possible to create a modified version of it that is well suited for NSP-based gene family identification.

Currently, work is in progress to create a customized version of the NCBI ORF Finder using perl scripts. This script reads a global alignment between two contigs and finds the longest common ORF between them. Earlier, a biologist could simply look at gaps inserted in an alignment and intuitively figure out whether a gap has caused a shift in codon positions. Information on such shifts is very important for accurately counting the first, second, and third position differences. It is possible to encode such intuitive rules into the customized ORF finding perl script.

The eventual goal is to create a computer based program that will read complete genomic data from various related organisms and will then identify and map all the genes and gene families. This is a lofty goal, but this thesis has proved that automating GI using evolutionary rationales takes us a significant step closer.

BIBLIOGRAPHY

Bie T, Cristianini N, Demuth J, Hahn M: *CAFE: A computational tool for the study of gene family evolution.* Bioinformatics Applications Note. 22(10), 1269–1271, 2006

Brown S, Chang J, Sadee W, Babbitt P: *A semiautomated approach to gene discovery through Expressed Sequence Tag data mining: discovery of new Human Transporter Genes.* AAPS PharmSci, 5 (1), 2003

Frank RL, Mane A, Ercal F: *An Automated Method for Rapid Identification of Putative Gene Family Members in Plants.* BMC Bioinformatics, 7 (Suppl 2):S19, 2006

Gaucher EA, De Kee DW, Benner SA: *Application of DETECTER, an evolutionary genomic tool to analyze genetic variation, to the cystic fibrosis gene family.* BMC Genomics, 7(44), 2006

Hekmat-Scafe D, Scafe C, McKinney A, Tanouye M: *Genome-wide analysis of the odorant-binding protein gene family in drosophila melanogaster*. Genome Res., 12, 1357-1369, 2002

Huang X, Madan A: *CAP3: A DNA Sequence Assembly Program.* Genome Research, 9(9), 868-877, 1999

Human Genome Program, U.S. Department of Energy: *DOE Human Genome Program Contractor-Grantee Workshop IV,* 1994

Liu Q: *Computational identification and systematic analysis of the ACR gene family in Oryza sativa.* Journal of Plant Physiology, 163(4), 445-451, 2006

Lockton S, Gaut BS: *Plant conserved non-coding sequences and paralogue evolution.* TRENDS in Genetics, 21:60-65, 2005

Nakano T, Suzuki K, Fujimura T, Shinshi H: *Genome-wide analysis of the ERF gene family in Arabidopsis and rice.* Plant Physiology (Rockville), 140(2), 411-432, 2006

Ohno S: *Evolution by gene duplication*. Springer-Verlag. ISBN 0-04-575015-7., 1970

Taylor JS, Raes J: *Duplication and Divergence: The Evolution of New Genes and Old Ideas.* Annual Review of Genetics 9: 615-643, 2004

Tian C, Wan P, Sun S, Li J, Chen M: *Genome-wide analysis of the GRAS gene family in rice and Arabidopsis.* Plant Molecular Biology, 54(4), 519-532, 2004

Yang Z, Nielson R: *Estimating synonymous and non-synonymous substitution rates under realistic evolutionary models.* Mol Biol Evol, 17:32-43, 2000

VITA

Cyriac Kandoth was born in Kerala, India on 23rd March, 1984. His primary education was spread out at various schools in Glasgow (Scotland, UK), New Delhi (India), and Trivandrum (Kerala, India). His secondary education in the city of Cochin, India, placed a focus on Physics and Computer Science. In 2005, he completed a Bachelor's degree in Computer Science and Engineering from Model Engineering College, Cochin (under the Cochin University of Science and Technology). After 6 months in the Software industry, Cyriac joined the University of Missouri - Rolla, USA (now known as Missouri S&T) and graduated with a Master of Science in Computer Science, with emphasis on Bioinformatics.

Cyriac Kandoth is currently a graduate student at UMR's Department of Computer Science. He is pursuing a PhD with a focus on Automated Gene Identification. His other research interests include Automated Surveillance Systems, Quantum Computing, Artificial Intelligence, Graphics Processing technologies, Parallel Processing, and pretty much anything that ends with –logy, -ysics, or –ience.