Pacific Symposium on Biocomputing 6:167-178 (2001)

NUCLEOTIDE SUBSTITUTIONS AND THE EVOLUTION OF DUPLICATE GENES

JOHN S. CONERY
Computational Science Institute
University of Oregon, Eugene, OR, USA

MICHAEL LYNCH
Ecology and Evolution Program
University of Oregon, Eugene, OR, USA

This paper describes software created to search for and analyze pairs of duplicate genes within a genome. The process is based on a program that uses aligned amino acid sequences to generate a corresponding alignment of the underlying nucleotide sequences and perform a codon by codon comparison of the nucleotides. Observed numbers of nucleotide substitutions can be used to make inferences about the ages of gene duplication events and the effects of natural selection acting on duplicate genes.

1 Introduction

Duplicate genes are found in a wide variety of organisms, including yeast[15], vascular plants[4], and vertebrates[12,13]. The existence of these duplicates has implications for comparative genomics, suggesting avenues of research based on searching for orthologous genes, and many diseases are often associated with gene duplications. The study of duplicate genes may also shed light on evolutionary processes; the double genome duplication hypothesis[12] is a conjecture that two rounds of complete genome duplication provided the "raw material" for morphological innovations that led to the establishment of the vertebrate genome.

Possible sources of multiple copies of a gene range from tandem duplication, which often leads to a copy of just a single gene, to chromosomal rearrangements that duplicate and rearrange larger sections of the genome, to polyploidization, in which the entire genome is duplicated. But what happens to duplicate genes that have become fixed in a population? Under what might be termed the classical model[12], there are two possible fates for a pair of duplicate genes: most likely one gene acquires deleterious mutations that cause it to be silenced, but in rare situations a mutation will be beneficial and one of the copies takes on a new function. Recently a third possibility has been suggested: mutations accumulate in both copies, partially degrading each gene until at some point the two genes have partitioned the functions of the original gene[2,11]. This new model, called duplication, degeneration, complementation (DDC), accounts for the fact that there are more functioning duplicates in extant organisms than can be explained by the classic model, and is also consistent with the observation that many genes have a modular structure that allows for independently mutable subfunctions.

We recently performed a series of computational experiments designed to shed light on the origin and subsequent evolution of duplicate genes.[10] For these experiments we constructed databases with the complete set of available coding sequences for nine different species. A heuristic similarity search program compared each sequence to every other sequence from the same genome to locate potential duplicates. We then subjected each pair to further analyses based on an examination of nucleotide substitutions.

This paper describes the software we developed for these experiments The next section is a description of a program that compares large numbers of pairs of genes. The input to the program is a stream of aligned amino acid sequences, and the output is a set of aligned nucleotide sequences with codons placed according to the amino acid alignment. We then describe how the nucleotide sequences can be processed to estimate the accumulation of nucleotide substitutions. We conclude by summarizing some of the results of our experiments.

2 A Genome-Wide Search for Duplicate Genes

Figure 1 is an overview of the flow of information in our search for duplicate genes. The starting point is a collection of protein sequences in Fasta format. We created a local database containing all the sequences from a genome, and then used BLAST[1] to compare each sequence to every other sequence from that genome. BLAST serves two roles in this process: it does the similarity search that locates pairs of similar genes, and it also produces an alignment of the amino acid sequences of those pairs. blast2align is a Perl script that extracts the sequence IDs, similarity score, and alignment for each pair found by BLAST.

The central application in this process is a program named ntalign. For each pair of sequences, ntalign uses the BLAST output as a guide to align the underlying nucleotide sequences so later steps in the process can examine the sequences codon by codon, for example to infer how many substitutions have occurred since the sequences diverged from a common ancestor (methods for estimating numbers of nucleotide substitutions are discussed in the next section).

By using the amino acid alignment to direct the nucleotide alignment we avoid frame shifts that would throw off the comparison of codons. For example, consider this fragment of an amino alignment and the corresponding nucleotide alignment generated when the BLAST output is used as a guide:

```
G-NGQ... ⇒   GGA---AATGGACAA...
GTNS-... ⇒   GGCACAAACTCA------...
```

If the nucleotide sequences are aligned directly, without respect to codon boundaries, an algorithm might produce a different alignment, one which has more matches and a smaller internal gap:

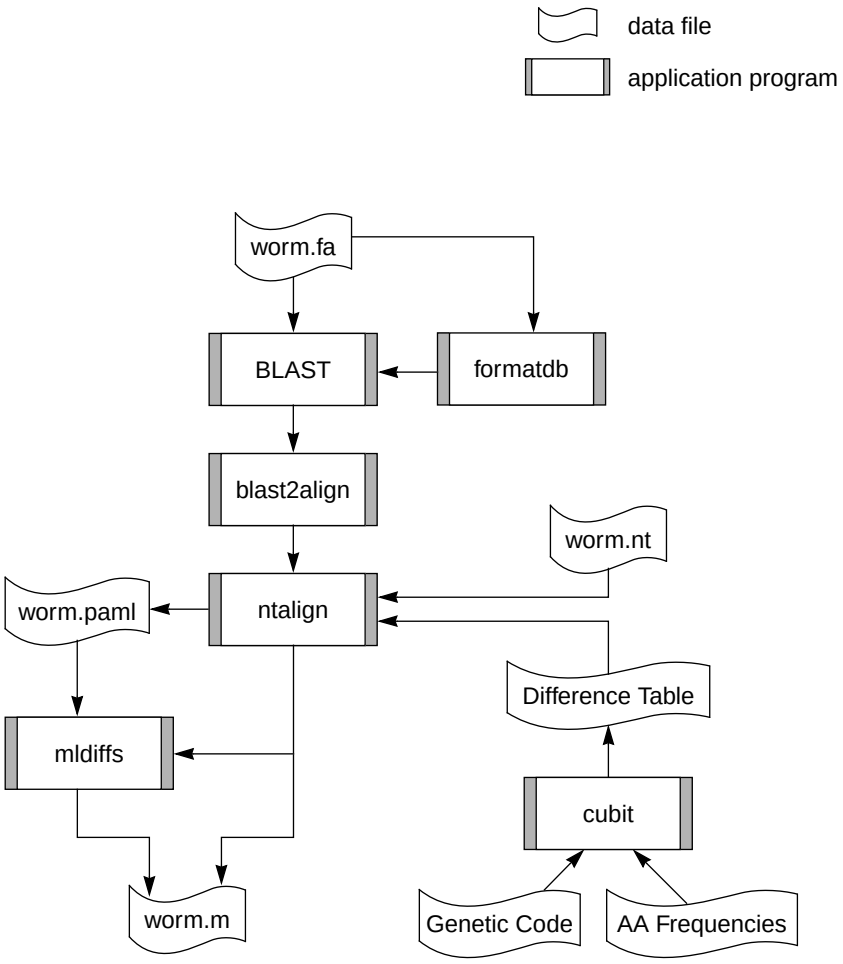

Figure 1: Analysis of Duplicate Genes. File names consist of the common name for a species and an extension that identifies the type of information: .fa is file containing a set of Fasta formatted amino acid sequences, .nt is a Fasta format nucleotide sequence file, .paml is a set of pairs of aligned nucleotide sequences, and .m is a Matlab format data containing data points and the visualization commands to display them. BLAST and formatdb are applications available from NCBI. blast2align and mldiffs are Perl scripts; ntalign and cubit are C++ applications. Source code for our Perl and C++ programs is available from the project web site (http://csi.uoregon.edu/projects/genetics/ntdiffs).

```
GG-AAATGGACAA...
GGCACAAACTCA-...
```

But this second alignment could not be used to analyze nucleotide differences in codons because the one-character gap introduces a frame shift and a codon-based analysis that processes the sequences three characters at a time will be misled.

Two parameters can be used to filter pairs to be analyzed. The program will ignore a match if the alignment score produced by BLAST is worse than a specified cutoff value. Users can also filter out sequences that have too many matches; for our experiments, we were interested in looking at duplicates that occur outside multigene families, so we analyzed pairs of genes only if both genes had five or fewer high-quality matches.

We were concerned that the amino acid alignments may have included some arbitrary placements on either side of a gap. Suppose the two sequences to be aligned are of the form *AXA* and *AZA*, where *A* designates a substring that has a good match in the other sequence, *X* and *Z* are substrings that do not match well, and *Z* is longer than *X*. During the alignment, one or more gaps will be inserted near *X*, resulting in one of these three patterns:

AX-A	*A-X-A*	*A-XA*
AVWA	*AUVWA*	*AUVA*

Here *Z* has been rewritten as the concatenation of strings *U*, *V* and *W*. If there is little or no information to guide the match between *X* and *Z* the placement of the gaps in the shorter sequence can be rather arbitrary; as shown above, the alignment algorithm may align X with the beginning, middle, or end of Z. To address this problem the ntalign application has an option that invokes a “gap expansion” algorithm. We define an anchor to be a site where both sequences have the same amino acid. Before aligning the underlying nucleotides, we look on each side of a gap for a region that begins with an anchor and in which two out of the seven amino acids (including the anchor) are an exact match. The probability of two out of seven random amino acids being the same is less than 5%. We then discard these regions so they are not included in the aligned nucleotide sequences.

A potential problem with this experimental design is the inclusion of pseudogenes and transposable elements in the data set. If a gene is silenced following duplication it can accumulate random mutations at any location, so including the comparison of pseudogenes with functioning genes can lead to a biased estimate of substitutions that affect coding sequences. One of the inputs to ntalign is a list of sequence IDs that should be ignored; these sequences are filtered out before counting the number of matches.

Not shown in Figure 1 are many other programs and data files that are an important part of the overall process. We made extensive use of the programs in the SEALS package[14] when creating the initial sequence database, controlling BLAST runs, and at other stages of the process. We wrote several additional Perl scripts that

Time			Diffs	Subst
t_0	ATGTATTCA	Original sequence		
t_1	ATGTATTCA ATGTTTTCA	Duplication	0	0
t_2	ATGTATTCA ATGT**T**TTCA	Single substitution	1	1
t_3	ATGTATTCA ATGT**C**TTCA	Multiple substitution	1	2
t_4	ATGTATTC**T** ATGTCTTCA	Single substitution	2	3
t_5	ATGT**C**TTCT ATGTCTTCA	Convergent substitution	1	4

Figure 2: Nucleotide Substitutions. Changes at each time period are indicated in boldface. The number of observed differences and the cumulative number of substitutions is given on the right.

interact with the SEALS programs to manage this process. Details of the complete process plus source code for our Perl scripts and C++ programs can be obtained from the project web site at http://www.csi.uoregon.edu/projects/genetics/ntdiffs.

3 Estimation of Numbers of Nucleotide Substitutions

As illustrated in Figure 2, the number of observed differences between two sequences is likely to be less than the number of substitutions that have occurred since the sequences diverged from a common ancestor. At time t_2 there has been one change, and the number of observed differences equals the number of substitutions. At t_3 there have been two substitutions, but the second occurred at the same site as the first so only one difference is observed. By t_5 two more substitutions have occurred, but one of them was a change that caused the two sequences to have the same nucleotide at an already modified site. The goal is to compute an estimate of the number of substitutions that have occurred as a function of the number of observed differences.

In protein-coding sequences, a substitution that does not lead to a change in the amino acid sequence is a *synonymous*, or *silent*, substitution. Mutations that cause a change in the protein are *nonsynonymous*, or *replacement* substitutions. For example, if the codon *GCC* changes to *GCA* the substitution is synonymous since both are codons for alanine, but the change from *GCC* to *GTC* is nonsynonymous because the amino acid generated by the new sequence will have a valine.

In a large-scale analysis involving several pairs of genes the sequences in one pair have a different length than sequences in another pair. Thus the goal is to compute the number of substitutions per site or number of substitutions per codon.

In our search for duplicate genes we used two different techniques for estimating numbers of substitutions, a method first introduced Li, Wu, and Luo[8] and later refined by Li,[6] and the maximum likelihood method of Goldman and Yang.[3] While maximum likelihood is the preferred method, it is much more expensive computationally, making it difficult to compare the hundreds of thousands of sequence pairs identified by BLAST. We implemented Li's method as part of the ntalign application and used it as an initial filter to identify pairs of sequences that should be examined more closely with maximum likelihood. The mldiffs program shown in Figure 1 is a script that selects pairs based on the initial estimates produced by ntalign, finds the corresponding aligned nucleotide sequences, and launches the maximum likelihood program[16] for that pair.

Li's method begins by characterizing positions within a codon as being nondegenerate, twofold degenerate, or fourfold degenerate, depending on whether zero, two, or four substitutions can be made at that site without changing the codon. For example, in the codon *AAT*, the first two sites are nondegenerate because any change is nonsynonymous. The third site is twofold degenerate: $T \Rightarrow C$ causes no change because *AAT* and *AAC* are both codons for asparagine, but $T \Rightarrow A$ and $T \Rightarrow G$ result in changes because *AAA* and *AAG* are codons for lysine.

Each observed difference is classified according to the type of site where it occurs and whether it is a *transition* (a change from purine to purine or pyrimidine to pyrimidine, i.e. $A \Leftrightarrow G$ or $C \Leftrightarrow T$) or a *transversion* (any other change). In this paper we use a vector notation when describing the number of sites and the number of transitions and transversions:

$N = \langle n_0, n_2, n_4 \rangle$	Number of sites of type $i \in \{0, 2, 4\}$.
$P = \langle P_0, P_2, P_4 \rangle$	Number of transitions at *i*-type sites.
$Q = \langle Q_0, Q_2, Q_4 \rangle$	Number of transversions at *i*-type sites.

The number of sites of each type and the numbers of observed transitions and transversions are used to compute two estimates of the number of nucleotide substitutions: the number of synonymous substitutions per synonymous site, denoted K_s, and the number of nonsynonymous substitutions per nonsynonymous site, denoted K_a.[6,8]

Two codons being compared may not have the same number of sites of each type. For example, consider a pair of sequences where one contains the codon *ACT* and the other contains *AAT*. *ACT* is a codon for threonine, and any change in the last position gives another codon for threonine, so this codon has two nondegenerate sites and one fourfold degenerate site: $N_{ACT} = \langle 2, 0, 1 \rangle$. *AAT* is a codon for asparagine. One change in the third site gives another asparagine codon, but the other two

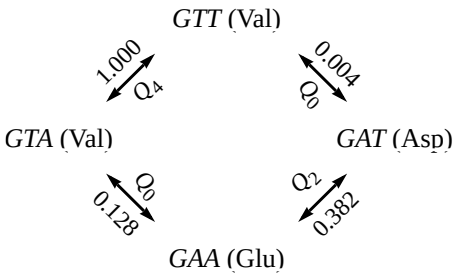

Figure 3: Evolutionary Paths. A graph showing transformations between *GTT* and *GAA*, assuming one mutation at a time. The edge labels show the relative frequency of the corresponding amino acid changes and the classification of the change as nondegenerate, twofold degenerate, or fourfold degenerate.

changes lead to lysine codons, so $N_{AAT} = \langle 2, 1, 0 \rangle$. To handle these situations, the algorithm for counting the number of sites of each type uses the average of the counts from each input sequence. So for this example,

$$N = (N_{ACT} + N_{AAT})/2 = \langle 2, 0.5, 0.5 \rangle$$

When there are two or three observed differences between a pair of codons the categorization of the differences depends on the order in which the changes evolved. As an example, suppose we are comparing two sequences where the codons are currently *GTT* and *GAA*. Suppose the ancestral sequence was *GTT* and as the result of two independent mutations this codon changed to *GAA* in one of the current sequences. If the change at the third site occurred first, the series of codons was $GTT \Rightarrow GTA \Rightarrow GAA$, but if the change at the second site occurred first the series was $GTT \Rightarrow GAT \Rightarrow GAA$. In the first series the change at the third site would be added to Q_4 because it is a transversion and the third site in *GTT* is fourfold degenerate. But in the second series the change at the third site would be classified as Q_2 because the third site in *GAT* is twofold degenerate. The difference at the second site is classified as Q_0 because changes in the second site are nonsynonymous in this case, and thus the two choices for counting the observed differences are $Q = \langle 1, 0, 1 \rangle$ or $Q = \langle 1, 1, 0 \rangle$.

Li et al[8] used a table of expected frequencies of amino acid changes to weight the different paths (Figure 3). The cumulative frequency of a series of mutations is the product of the individual frequencies along each edge. In the example, one path from *GTT* to *GAA* has a frequency of $1.0 \times 0.128 = 0.128$, and the other path has a frequency of $0.004 \times 0.382 = 0.015$. The relative probabilities are thus

$$0.128/(0.128 + 0.015) = 0.893$$

for the series in which the transversion occurred at the fourfold degenerate site and

$$0.015/(0.128 + 0.015) = 0.107$$

for the series in which the transversion occurred at the twofold degenerate site. The relative probabilities are used to weight the transition and transversion vectors, so for this pair of codons the observed differences are $Q = \langle 1.0, 0.107, 0.893 \rangle$.

The previous example assumed one of the current codons was the ancestral

codon and the mutations both occurred in the same sequence. But the relative path weights do not depend on the ancestral state; as long as we assume the differences arose through two independent mutations the calculation of the relative weights does not depend on which vertex is used as the starting point.

In order to make genome-scale sequence comparisons, ntalign uses a 64×64 table of precomputed *N*, *P*, and *Q* vectors, so the algorithm that counts and classifies differences simply has to do a table lookup using two codons as table indices, compute the sum of *N*, *P*, and *Q* over all codons, and then calculate K_s and K_a using the vector sums.

Our method for counting and classifying the differences between pairs of codons is based on a data structure we call a *difference cube*, an *n*-dimensional cube in which each vertex is labeled by a codon. It is basically an *n*-dimensional binary hypercube, except node labels are 3-letter codon strings instead of *n*-bit binary numbers. The labels are arranged so adjacent nodes are labeled by codons that differ at exactly one site.

For any pair of codons, the number of dimensions in the corresponding difference cube is defined by the number of differing sites. The example in Figure 4 shows the 3D difference cube for the pair of codons *GTT* and *AAA*. Note that the two differences between *GTT* and *GAA* used in the example of Figure 3 are represented in one of the faces of the 3D cube.

The following algorithm, based on a common algorithm for creating a binary hypercube[5], will create the difference cube for a pair of codons *X* and *Y*:

```
d = 0;
cube = node(X);
for i = 1..3:
    if X_i ≠ Y_i
        d = d+1
        expand(cube,i,Yi)
```

The number of dimensions *d* is initialized to 0, and the cube is initialized as a single node labeled by one of the codons (the notation *node(X)* means "create a new node with label *X*"). The loop iterates over each of the three sites. At a site where the two

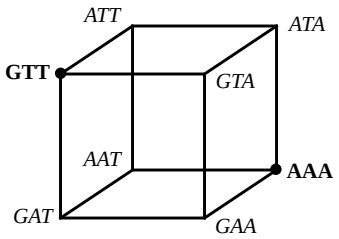

Figure 4: Difference Cube for GTT and AAA. Since this pair of codons has differences at all three sites the resulting cube has three dimensions. The "front" face of this cube is the 2D cube (i.e. square) for the pair of codons *GTT* and *GAA*.

codons differ, the cube is expanded by making a complete duplicate of each current node, adding an edge between each existing node and the corresponding node in the copy, and changing the i^{th} position in the label in each new node to be Y_i. When the loop terminates d will be the dimension of the resulting cube.

From the n-dimensional difference cube for a pair of codons it is clear there are $n!$ unique paths, each of length n, between the two vertices for the codons. Each step on a path traverses a different dimension. Starting from one codon, there are n choices for the first step, $n-1$ for the second step, down to one choice for the last edge. The paths can be enumerated by creating all permutations of the list of dimensions $1 \ldots d$.

Using the above method for constructing a difference cube for a pair of codons, the table of codon differences is built with the following algorithm:

```
for X ∈ {AAA, AAT, …, GGG}
    for Y ∈ {AAA, AAT, …, GGG}
        C = difference_cube(X,Y)
        assign_frequencies(C)
        N_X,Y = differences(X,Y)
        for path ∈ permutations(1…dim(C))
            w = w + weight(path)
        for path ∈ permutations(1…dim(C))
            P_X,Y += transitions(path,w)
            Q_X,Y += transversions(path,w)
```

After constructing the difference cube and assigning expected frequencies to each edge in the cube we compute the sum of relative frequencies of all the paths. Then each path is traversed again, this time counting the number of transitions and transversions along the path and weighting the counts as a fraction of the total weights.

Stop codons are not usually included in comparisons of coding sequences, but they can occur as intermediate nodes. For example, when counting the differences between *AAA* and *TAT*, intermediate states are *AAT* (asparagine) and *TAA* (stop). We assign an expected frequency of 0 to any edge connected to a vertex labeled with a stop codon, which leads to a cumulative frequency of 0 for any path that includes this vertex. The net effect is that paths through stop codons are not included in the path weight calculation.

4 The Evolution of Duplicate Genes

We used the process outlined in Figure 1 to analyze gene duplicates in a variety of model species, including brewer's yeast (*Saccharomyces cerevisiae*), nematode (*Caenorhabditis elegans*), thale cress (*Arabidopsis thaliana*), and several vertebrates, including human (*Homo sapiens*) and mouse (*Mus musculus*)[10]. After using BLAST

to identify pairs of similar amino acid sequences, we ran ntalign to process the underlying nucleotide sequences and calculate K_s and K_a for each pair. We then used the maximum likelihood method to recompute the estimated numbers of substitutions for those pairs with $K_s < 99$. To distinguish between the estimates computed by each program, we will use Yang's notation and refer to the maximum likelihood estimates of the number of synonymous substitutions per synonymous site as d_S and nonsynonymous substitutions per nonsynonymous site as d_N.

4.1 Age Distribution of Duplicate Genes

If we assume the number of synonymous substitutions per synonymous site is proportional to the age of a gene duplication event we can plot a histogram of d_S values to get the age distribution of duplicate genes. In all species, most duplicates appear to be quite young, with $d_S < 0.1$.[10]

There is an interesting peak in the age distribution of *Arabidopsis* genes at $d_S = 0.8$, indicating there are several pairs of duplicate genes of a similar age. This peak might be evidence in support of polyploidization in *Arabidopsis*.[4] Assuming an estimated silent site substitution rate of 6.1 mutations per site per million years[7,9] we date this peak at about 65 million years ago.

Our analyses of vertebrate genomes did not show such a peak. If there are a large number of duplicates remaining from a double genome duplication before the divergence of ray-finned and lobe-finned fishes 430 MYA, a silent-site substitution rate of 2.5 mutations/site/MY[7] might lead to a peak in the age distribution at around $d_S = 2.2$. However, it would be difficult to make any conclusions based on such a high value of d_S.

4.2 Relaxed Selection Following Duplication

By plotting d_N as a function of d_S we can get an overall view of how natural selection acts on duplicate genes. Again assuming d_S is a measure of the age of a pair of duplicates, a value of $d_N < d_S$ implies selection against amino acid changing substitutions, $d_N = d_S$ indicates relaxed selection, and $d_N > d_S$ means selection for protein evolution. Both the classic model and the DDC model described in the introduction predict a period immediately following a duplication when the genome should be able to tolerate a high degree of nonsynonymous substitutions in one member of a duplicate pair because the other member is still functioning at full strength.

Figure 5 shows a log-log plot of d_N vs. d_S for *C. elegans*. The diagonal is a plot of $d_N = d_S$, so points near the line are pairs less affected by selection against amino acid changes. In this plot, and in the plots for all the other species we studied, there are some pairs above the diagonal at small d_S values, but most are below, implying

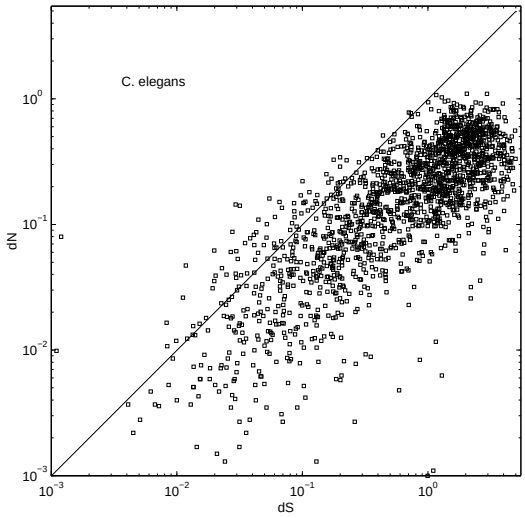

Figure 5: Duplicate Genes in *C. elegans*. A plot of the number of nonsynonymous substitutions per nonsynonymous site (d_N) vs. the number of synonymous substitutions per synonymous site (d_S) for pairs of *C. elegans* genes. Points below the diagonal ($d_N < d_S$) imply the genes have been subjected to purifying selection against amino acid changes.

that almost as soon as they arise gene duplications are subjected to selection pressure.[10]

5 Summary

This paper described a method of searching for and analyzing pairs of duplicate genes within a genome. At the heart of the process is a computer program that uses aligned amino acid sequences to generate the corresponding alignment of the underlying nucleotide sequences in order to compare codon differences. The program includes an efficient table-driven implementation of a method for estimating numbers of nucleotide substitutions; values computed by this approximate method can be used to identify pairs that should be examined more closely. Numbers of substitutions can be used to make inferences about the ages of gene duplication events and the effects of natural selection acting on duplicate genes. Other analyses, including stochastic models of the birth and death rates of duplicate genes and a discussion of a potential role played by gene duplication in speciation, can be found in our forthcoming paper.[10]

References

1. Altschul, S.F. et al. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 25:3389-3402.

2. Force, A., Lynch, M., Pickett, F.B., Amores, A., Yan, Y.L., and Postlethwait, J. 1999. Preservation of duplicate genes by complementary, degenerative mutations. *Genetics* 151(4):1531-45.

3. Goldman, N, and Yang, Z. 1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. Mol. Biol. Evol. 22:160–174.

4. Grant, D., Cregan, P., and Shoemaker, R.C. 2000. Genome organization in dicots: Genome duplication in *Arabidopsis* and synteny between soybean and *Arabidopsis*. *Proc. Nat. Acad. Sci. USA* 97:4168–4173.

5. Leighton, F. T. 1992. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*. Morgan Kauffman, San Mateo, CA.

6. Li, W.-H. 1993. Unbiased estimation of the rates of synonymous and nonsynonymous substitution. *J. Mol. Evol.* 36:96-99.

7. Li, W.-H. 1999. *Molecular Evolution.* Sinauer Associates, Sunderland, MA.

8. Li, W.-H., Wu, C.-I., and Luo, C.-C. 1985. A new method for estimating synonymous and nonsynonymous rates of nucleotide substitution considering the relative likelihood of nucleotide and codon changes. *Mol. Biol Evol.* 2(2):150-174.

9. Lynch, M. 1997. Mutation accumulation in nuclear, organelle, and prokaryotic transfer RNA genes. *Mol. Biol. Evol.* 14:914–925.

10. Lynch, M. and Conery, J.S. 2000. The evolutionary fate of duplicate genes. *Science* (accepted for publication).

11. Lynch, M, and Force, A. 2000. The probability of duplicate gene preservation by subfunctionalization. *Genetics* 154(1):459-73.

12. Ohno, S. 1970. *Evolution by Gene Duplication.* Springer-Verlag, Heidelberg, Germany.

13. Sidow, A. 1996. Gen(om)e duplications in the evolution of early vertebrates. *Curr. Opin. Genet. Devel.* 6:715-722.

14. Walker, D.R., and Koonin, E.V. 1997. SEALS: A System for Easy Analysis of Lots of Sequences. *Intelligent Systems for Molecular Biology* 5:333-339.

15. Wolfe, K. H., and Shields, D.C. 1997. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature* 387:708-713.

16. Yang, Z. 1997. PAML: a program package for phylogenetic analysis by maximum likelihood. *Comput. Appl. Biosci.* 13(5):555-6.