CS#594 - Group 13 (Tools and softwares)

By, Murali Sivaramakrishnan, Ognjen Perisic,, Shashi Ranjan.

Abstract:

The protein and DNA sequence library are growing very rapidly, almost 50% per year. The major problem faced by the biologists, is the need to compare one sequence against all the sequences in the database in a reasonable time. A newly discovered sequence is compared to the known sequences in the database. Often this search provides the first insight into the mechanism of action of a new sequence. Many softwares and tools have been developed for this purpose. Each having its own trade off in terms of selectivity, sensitivity and speed. We here in this paper present few of them (FASTA, BLAST, Gapped BLAST, PSI-BLAST, Pip Maker, MACAW, Motif Explorer and the five tools for finding conserved sequences in multiple alignment)

1.0 Introduction:

The DNA and Protein sequences are the blueprint for an organism's structure and function. All the information about the living world is encoded as different sequences in these units. Analyzing them gives us lots of information regarding the various biological aspects of life and their evolution.

The protein and DNA sequence library are growing very rapidly, almost 50% per year, owing to the more efficient cloning techniques and more productive sequence procedures. Some of the major DNA & protein databases are the EMBL, GenBank, DDBJ, Prosite. The importance of the database is to provide current view of know sequences quickly and to learn new biology. A newly discovered sequence is compared to the known sequences in the database. Often the matched sequences obtained from a database search provide the first insight into the mechanism of action of a new sequence.

Now equipped with this vast amount of data in the form of sequences, the foremost problem on hands of the biologists is the need to compare one sequence against all the sequences in the database in a reasonable time. This problem can be dated back to 1980's, but then the database ware small. The Dynamic Programming methods for comparing two sequences, more precisely, the alignment of two sequences (global or local) allows to obtain an optimal alignment in time proportional to the product of the lengths of the two sequences, i.e. when applied to the entire database the computational time would grow linearly to the size of the database. At present time, the size of a database is much greater and the approach of Dynamic Programming will be impractical for searching large databases unless we use supercomputer or some parallel processors [2]. This was the motivation behind developing new algorithms and software tools. A brief overview of some of the algorithms and tools used are given in the following paragraphs.

David Lipman and William R. Pearson in 1985 described the **FASTP** program for searching protein sequence libraries. FASTP combines a rapid technique for focusing on those regions in a pair of sequence that share a high density of identities with a scoring procedure that uses the PAM250 scoring matrix for high sensitivity. It decreased the typical search time from 6hours to 10 min. The **FASTA**, is a resent improved version of FASTP. It combines the function of the FASTP and FASTN (modified version of FASTP for DNA sequence). It along with calculating this initial score also checks to see if several initial regions can be joined together in a single alignment to increase the initial score, thus improving sensitivity. In addition, the **FASTA** package includes programs fro comparing a protein sequence to a DNA sequence database (TFASTA), for identifying local sequence similarities or duplications in sequences (LFASTA, PLFASTA), and for evaluating the statistical significance of a similarity score (RDF2).

The **BLAST** software or program is a set of comparison algorithms introduced in 1990 by S.F.Altschul that employs a measure based on mutation scores [2]. The results are directly approximated as opposed to dynamic programming. These are used to search sequence databases for optimal local alignments for a given query sequence.

The main strategy of BLAST is to quickly seek or locate un-gapped similar regions between the query sequence and sequences from the database. Given a cutoff score *S* as parameter, The BLAST first generates all the words of length w formed with the alphabet of the sequences and then each word of the query is compared with each word of this set and a similarity threshold T is recorded. It then selects only those words that contain a word pair with a score of at least T, thus generating a list of words of length w whose score is more than T. The next step is to compare every word of the list with score atleast equal to T, with each word in the database. All exact matches with DB sequences detected are called hits. Every hit

is then extended in both directions to determine whether this hit is part of a larger segment of similarity i.e. if it is contained within a segment pair whose score is greater than or equal to *S*. Every such hit is retained and called an HSP (high scoring segment pair) and the best scoring HSP is the MSP (maximal segment pair) is reported.

In **Gapped BLAST**, as the name says, gapped alignments have been introduced to find alignments in a sequence that are with in some limited value instead of finding several un gapped alignments from the sequence. This essentially increases the initial database scan, but we need to locate only one of the essential HSPs in order obtain a success. The second refinement is the "two – hit" method, were extensions are made only if there are two non overlapping word pairs within some distance A. The advantage of this is that the program executes three times faster because only a few hits are extended [4].

Another version of BLAST is the **PSI-BLAST** for position specific iterated BLAST, which involves a series of repeated steps. First, a database search of a protein sequence database is performed. The results of the search are visually inspected to see if any sequence from the database is related to the query sequence. If so then the search is repeated again. The scoring sequence matches found from the first iteration are aligned. From this alignment a sequence motif (short amino acid sequence), which indicates the differences at each position of the alignment is produced. The database is searched again with this motif, thus expanding the search to include sequences that match the variations found in the motif at each sequence position. The results, which are obtained, indicate that new sequences are found which are more significantly related to the motif than the previously found sequences.

Pipmaker is a WWW site for comparing two long DNA sequences to identify conserved segments and for producing informative, high-resolution displays of the resulting alignments. One display is a percent identity plot (pip), which shows both the position in one sequence and the degree of similarity for each aligning segment between the two sequences in a compact and easily understandable form. Positions along the horizontal axis can be labeled with features such as exons of genes and repetitive elements, and colors can be used to clarify and enhance the display. Another display is a plot of locations of those segments in both species (similar to a dot-plot). PipMaker is appropriate for comparing genomic sequences from any two related species, although the types of information that can be inferred depend on the level of conservation and the time and divergence rate since the separation of the species. PipMaker is capable of returning the alignments generated by BlastZ in any or all of four different formats: a pip, a dot plot, a conventional textual alignment, and a compact listing of the coordinates of the aligning segments.

Multiple Alignment Construction & Analysis Workbench (MACAW) is an interactive program that allows you to to construct multiple alignments by locating, analyzing, editing, and combining "blocks" of aligned sequence segments. MACAW incorporates several novel features. (1) Regions of local similarity are located by a new search algorithm that avoids many of the limitations of previous techniques. (2) The statistical significance of blocks of similarity is evaluated using a recently developed mathematical theory. (3) Candidate blocks may be evaluated for potential inclusion in a multiple alignment using a variety of visualization tools. (4) A user interface permits each block to be edited by moving its boundaries or by eliminating particular segments, and blocks may be linked to form a composite multiple alignment.

Motif explorer is a software tool, which uses indexing structure based on generalized suffix trees for high performance motif searches, and makes the interactive motif exploration possible. It's fast enough to provide answers within seconds. It combines the implementation of large-scale GST structures with the GST regular expression search algorithm. The GST is created and processed as linear steams of data, using limited main memory and without random access to the entire structure. The motif explorer uses the GST regular expression search performance predictor to improve the performance of the search. It has a World Wide Web (WWW) based interface and presents the results as HTML docs. It also gives the WWW links to other appropriate database containing information of the matching sequences. The motif Explorer, in its average and median search timings, is less by approximately an order of 20 when compared to the grep program.

Another important problem for molecular biologists and computer scientists is distinguishing functional regions from those whose similarity reflects the residual common ancestral sequence that has not yet changed via evolutionary drift. Here will be presenting five of them.

Two of **five methods** described in this part are aimed at finding protein-binding sites on DNA. Such sites are usually a series of consecutive positions, one or more of which can vary somewhat without measurably changing the binding affinity. Thus it is desirable to examine a series of neighboring positions in each row when finding blocks. Each row-based method allows up to **k** mismatches per row; in one

method the mismatches are relative to a specified 'center' sequence (e.g. the human sequence) and in the other the mismatches are relative to an unknown 'center' sequence.

Certain parameters are common to all of the tools/programs/algorithms. The minimum length of the regions to be reported and the minimum number of sequences, which must be active, are selectable by the user. The search can be conducted in the entire alignment or it can be restricted to a portion specified by a given range in any of the sequences. The **Five tools** are

agree : This utility locates regions in a given alignment that have good column agreement.
infocon : The length of the region is often a reliable indicator that some functionality was preserved across the species, but conservation doesn't need to be perfect and such regions might be fragmented into conserved pieces too small to be detected, so a systematic way to link the smaller regions is needed. The two utilities **infocon** and **phylogen** are trying to solve this problem. The idea is to assign a numerical score to each column and then look for runs of columns meeting the following two conditions:

1. their cumulative score (obtained by adding together the individual column scores) is no smaller than the score of any of their sub-runs;
2. they are maximal with this property, i.e. they are not contained in any longer run having the property 1.

The **infocon** tool finds full runs of columns with high information content in the given alignment.
phylogen : This program scores the columns by their evolutionary relationships among the sequences of the given alignment implied by a supplied phylogenetic tree. The phylogenetic tree has a leaf node for each species and each internal node represents a putative common ancestor for the species in its sub-tree.
kkno : This program scans the alignment to determine, starting at each position, the longest region in which no row differs from a specified, known 'center' sequence in more than **k** positions.
kunk : This program is similar to **kkno** except that the center sequence is not known a priori; instead, the program computes the 'best' center sequence for each conserved region it finds. This center sequence can be thought of as belonging to a common ancestor of the species represented in the alignment or as a potential binding site for known or unidentified proteins.

2.0 Rapid and Sensitive Sequence Comparison with FASTP and FASTA

In 1985, David Lipman and William R. Pearson described the FASTP program for the searching proteins sequence libraries. FASTP combined a rapid technique for focusing on those regions in a pair of sequence that, share a high density of identities with a scoring procedure that uses the PAM250scoring matrix for high sensitivity. FASTP decreased the computer time required for a protein database search, from about 6 hours on a vax11/780 t about 10min on an IBM-PC. FASTA is the new improved version of the FASTP. It combines the functions of the FASTP and FASTN programs and provides a more sensitive sequence comparison algorithm. The FASTA package includes other programs like TFASTA for comparing the protein sequence to a DNA sequence database LFASTA and PLFASTA for identifying local sequence similarities or duplications in sequence and RDF2 for evaluating the statistical significance of a similarity score. The FASTA programs can be tailored to specific comparison problems by changing the similarity scoring matrix and gap penalties. I.e., the same program could be used to compare protein sequence with PAM250 matrix and a matrix based on genetic codes to compare DNA. FASTA Program also has many output options that can be used to highlight similarities and differences in aligned sequences. In any database search there is always a sequence with the best score, regardless of weather that sequence share a common ancestry or some other similarity with the query sequence. In any sequence comparison there is a trade-off between sensitivity- the ability to identify related sequence and selectivity- the avoidance of false positives (unrelated sequences with high similarity scores. In many cases , the main problem is to differentiate between high scoring sequences that share common ancestry or significant similarity from those which don't but have high scores due to local sequence compositions and random chance.

The FASTA program is more sensitive than the FASTP, but with the increase sensitivity the selectivity decreases thus and additional care is required when interpreting the results of a FASTA search.

Using the FASTA Programs:

Although the FASTA Programs provide a number of options for customized searches, the most of the time only three entries are required: the name of the file containing the query sequence, the name of the

file containing the library of the second sequence and the value of the ktup parameter. FASTA and TFASTA compare the query sequence to all the sequences in the second file, there need only be one, and report the one best similarity score and alignment for each pairwise comparison. All of the FASTA programs calculate a best local similarity score. The score of the local region is not affected by poorly aligned portions of the sequences outside the best region. Thus, programs in the FASTA package can be used to find conserved or shuffled proteins domains.

FASTA Implementation:

The programs are written in C programming language. The program code is very portable, exactly the same source code compiles on all the machines except the Macintosh. The complete source code is available for all versions of the program from William R. Pearson. The searching programs FASTA and TFASTA can search libraries in a variety of different formats, including: 1) FASTP/DM or query sequence format; 2) GenBank magnetic tape format; 3) the protein Identification resource CODATA format 4) EMBL and SWISS-PROT format 5)IntelliGenetics sequences file format and 6) compressed GenBank format for floppy disk distribution.

The behavior of all the FASTA programs can be modified by specifying a different scoring matrix or CUTOFF values, and the alignments displayed by FASTA, TFASTA and LFASTA can be modified by specifying options on the command line or with environment variables.

FASTA Algorithm:

FASTA uses four steps to calculate three scores that characterize sequences similarity. The first step uses a rapid technique for finding identities shared between two sequences. The method is similar to the technique described by Wilbur and lipman. FASTP and FASTA achieve much of their speed and selectivity in this first step by using a lookup table to locate all identities or groups of identities between two DNA or amino acids sequences. A lookup table is a rapid method for finding the position of a residue in a sequence. One way to find the “A” in a sequence “NDAPL” is to compare the “A” to each residue in the sequence. A faster way is to make a table of all possible residues (23 for proteins) so that the computer representation of the residue is same its position in the table. A value us then placed in the table that indicates whether the residue is present in the sequence and of it is, then at what location.

The ktup parameter determines how many consecutive identities are required in a match. A ktup value of 2 is frequently used for protein sequence comparison, which means that the program examines only those portions of the two sequences that have atleast two adjacent identical residues. The sensitivity of the search rises with the decrease in the ktup value and the selectivity of the search decreases with the decrease in ktup value. Thus a tradeoff between selectivity and sensitivity is need here. For DNA sequence comparisons, the ktup value ranges from 1 to 6; values between 4 and 6 are recommended.

A sequence comparison using the traditional dynamic programming technique requires a number of residue comparisons proportional to the product of the lengths of the sequence being compared, for example to compare a hemoglobin beta chain (146 amino acids) with trypsin (229 amino acids) nearly 33,434 comparisons are required but with the FASTP or FASTA only 94 comparison are required (ktup parameter =2) or 1921 comparisons with ktup parameter =1.

In addition to the lookup table the algorithm also uses the “diagonal” methods to find all regions of similarity between the two sequences, counting ktup matches and penalizing for intervening mismatches. This method identifies regions of a diagonal that have the highest density of ktup matches. The term diagonal refers to the diagonal line that is seen on a dot matrix plot when a sequence is compared with itself. And it denotes and alignment between two sequences without gaps.

FASTP uses a simple formula to identify portions of a diagonal with a high density of identities, referred to as a local region of similarity or simply a region. FASTA uses a formula for scoring ktup matches that incorporate the actual PAM250 values for the aligned residues. Thus groups of identities with high similarity scores contribute more to the local diagonal score than do identities with low similarity scores. For DNA sequences comparison uses a constant value for ktup matches. FASTA saves the ten best local regions, regardless of whether they are on the same or different diagonals.

After the ten best local regions are found in the first step, they are rescored using a scoring matrix that allows runs of identities shorter that ktup residues and conservative replacements to contribute to the similarity score. For Protein sequences, this score is usually calculated using the PAM250 matrix. For DNA sequence comparisons, the matrix can be constructed that allow separate penalties for transitions and

transversions. For each of the best diagonal regions rescanned with the scoring matrix, a sub region with the maximal score is identified.

The FASTP Program uses the single best scoring initial region to characterize pair-wise similarity; the initial scores are used to rank the library sequences. The FASTP initial score is also calculated by FASTA, and it is referred to as the init1 score. FASTA goes one step further during the library search; it checks to see whether several regions can be joined together in a single alignment to increase the initial score. This FASTA improves on the sensitivity of FASTP by allowing multiple high scoring initial regions to be joined. Given the locations of the initial regions, their respective scores, and a "joining" penalty (analogous to the gap penalty), FASTA calculates an optimal alignment of initial regions as a combination of compatible regions with maximal score. This optimal alignment of initial regions can be rapidly calculated using a dynamic programming algorithm. FASTA uses the resulting scores referred to as the initn score, to rank the library sequence. The third "joining" step in the computation of the initial score increases the sensitivity of the search method because it allows for insertions and deletions as well as conservative replacements. The modification, however, decreases selectivity. The degradation is limited by including in the optimization step. Only those initial regions whose scores are above an empirically determined threshold. After the complete search of the library, FASTA plots the initial scores of the each library sequence in a histogram, calculates the mean similarity score for the query sequence against each sequence in the library, and determines the standard deviation of the distribution of the initial scores. The highest scoring library sequences are aligned using a modification of the standard MWS algorithm. The optimization employs the same scoring matrix used in determining the initial regions, the resulting optimized alignments are calculated for further analysis of potential relationships, and the optimized similarity score is reported.

FASTA and TFASTA report only the similarity score of the one best pair wise alignment between two sequences. In the case of proteins with repeated domains, there may be several alignments with high similarity scores that are of biological interest. Multiple regions of similarities can be displayed as alignments or as a dot-matrix style plot by LFASTA and PLFASTA, respectively. LFASTA and PLFASTA use a slightly modified FASTA algorithm to focus more tightly on local alignments.

RDF2 is designed to evaluate the statistical significance of a pair wise similarity score. The program calculates similarity scores for the best pair wise alignment using the FASTA algorithm, then randomly shuffles the second sequence and calculates pair wise scores for the query sequence and the shuffled sequence. By examining the distribution of similarity scores obtained one can evaluate the likelihood that the similarity score for the unshuffled sequence are due to unusual sequence composition or other random fluctuations.

TFASTA is a program for comparing a protein sequence to a DNA sequence by translating the DNA sequence in all six reading frames. The value if doing sequence searches with proteins rather than DBA sequences cannot be overemphasized. Since TFASTA compares a protein to a DNA sequence in all six reading frames, it can be used to check for frameshifts in cDNA sequences when other homologs for the protein coded by the cDNA are known.

FASTA was originally designed to search protein sequences libraries for homologus sequences. But it can be used to search DNA and short query sequences too. Few easy adjustments are required to tune it for such searches. For short query sequences the ktup should be equal to one and the CUTOFF threshold should be lowered. For DNA the SMATRIX variable to the environment of the program execution could be made to point to the appropriate matrix file. FASTA also have a variety of input and output option. The output options control how much of the sequence alignment are shown and how identities are highlighted.

FASTA is both fast and selective and its sensitivity has been increased not only by using the PAM250 matrix but also by joining initial regions. When compared to the NWS based programs FASTA is slightly less sensitive but considerably more selective and 100 times faster.

The FASTA Program can search the NBRF protein sequence library (2.5 million residues) in less that 20 min on an IBM-PC microcomputer and unambiguously detect proteins that shared a common ancestor billions of years in the past.

3.0 Basic Local Alignment Search Tool (BLAST) and its extensions Gapped BLAST and PSI-BLAST:

Searching for similarity between two biological sequences and comparing these sequences is the fundamental means by which we can understand evolution. There are many tools in bio-informatics developed to accomplish this task. The most popular or widely used tool is BLAST (Basic Local Alignment Tool). Stephen Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman developed the BLAST programs at the National Center for Biotechnology Information (NCBI). In the next few sections we will discuss the principles, workings, application and pitfalls of BLAST. Then we will see what extensions are made to the original BLAST to over come its disadvantages.

Before we can compute the similarity of tow sequences, their proper alignments must be determined, which leads to the following questions: What is the best alignment between two sequences? How should the alignments be scored? When evaluating a sequence alignment, one would like to know how meaningful it is. This requires a scoring matrix or a table of values that describe if there is a match or a mismatch between the residue-pairs of amino acid or nucleotide occurring in an alignment. In the case of comparing two nucleotides sequences, all that is being scored is whether or not two bases at a given position are the same. But with protein sequences the situation is different, because the scoring matrices or the substitution matrices for amino acids are more complicated [2].

There are two types of substitution matrices, which are commonly used and they are the blocks substitution (BLOSUM) [2] and point accepted mutation (PAM) [2]. Although both are based on taking sets of high-confidence alignments of many homologous, similar originating proteins and assessing the frequencies of all substitutions, they are computed using different methods, which we will not discuss them here. For our purposes, we will just refer to either one of these scoring or substitution matrices.

As mentioned before, there are a number of tools or software available for searching sequences in the databases. Why do we need an automated way of finding the optimal alignment and why BLAST became the number one choice amongst biologists? It is clear that choosing a good alignment by eye is possible, but the procedure cannot be repeated more that once or twice. Also it is apparent that to guarantee best alignment, many alignments must be generated and evaluated. Before BLAST was introduced, Dynamic Programming methods of Needleman and Wunsch [2] and Smith-Waterman [2] were being used to find the optimal solution.

The method of dynamic programming does produce optimal results but are not practical when the search is after a database with many, many sequences. This is where BLAST comes in, which employs a measure based on well-defined mutation scores. BLAST directly approximates or guesses the results that would have been obtained by dynamic programming algorithm for optimizing this measure [1]. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, making real matches easier to distinguish from random background hits [4]. BLAST uses a heuristic algorithm, which seeks local as opposed to global alignments and is therefore able to detect relationships among sequences that share only isolated regions of similarity [1]. However, the similarities detected will be weak as compared to dynamic programming, but it will still be a significant result for biologist and faster too.

Before we look at the operations involved in BLAST, lets define some terms: a segment is contiguous subsequence of a nucleotide or amino acid sequence, a segment pair is a pair of segments of the same length, one from each of the two sequences being compared. Given this the basic task that BLAST performs is that it identifies all pairs of similar segments whose score is greater than a given threshold T. The resulting pairs of similar segments are called high-scoring segment pairs (HSPs). The segment pair with the highest score is the maximal-scoring segment pair (MSP), i.e. the highest scoring pair of identical length segments chosen from two sequences.

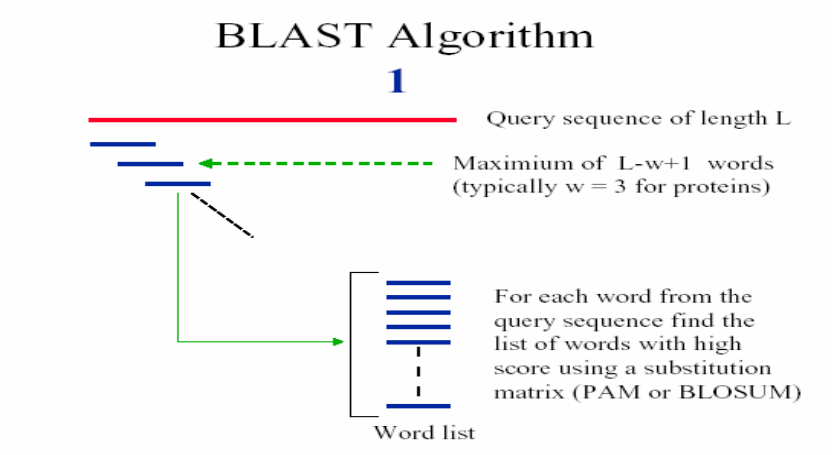
The BLAST algorithm is a heuristic for finding locally optimal sequence alignments that was developed by the National Center for Biotechnology Information at the National Library of Medicine [5]. BLAST is actually a set of similarity search programs designed to search all the available sequence databases, regardless of whether the sequence query sequence is DNA or protein. To see the different BLAST search programs refer to [5].

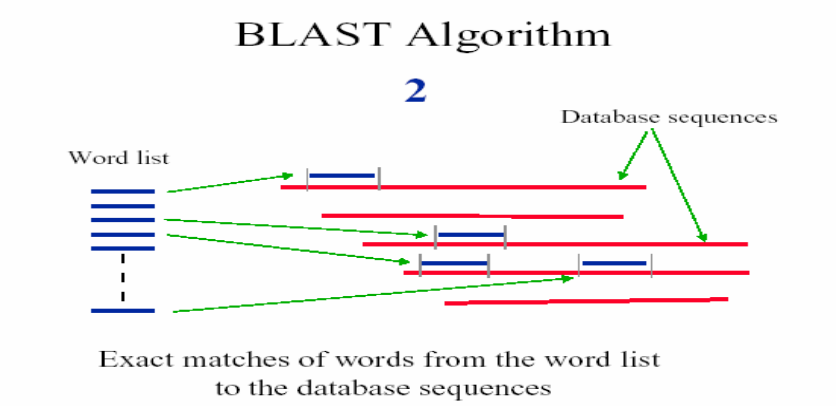
When comparing two sequences, BLAST performs the following three major steps or algorithms (see the figure given below):

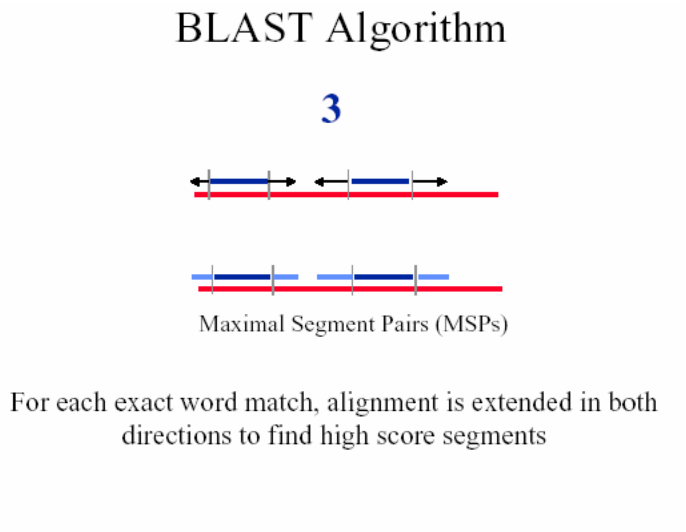
1. Given a query sequence and a word length w, BLAST generates a list of all words that make up the query. By word we mean a contiguous segment of one of the sequences that has a pre-determined length. The default word lengths are usually 3 for proteins and 11 for nucleotides. This length is the minimum needed to achieve a word score that is high enough to be significant but not so long as to miss short but significant patterns. Then BLAST uses a scoring matrix, Blosum62 for proteins, to determine all high-scoring matching words. A list of all words (w-mers) that can score greater than the given threshold, called the neighborhood word-score threshold, when compared to w-mers from the query is created.

2. The database is searched using the list of w-mers generated in step one to find the corresponding w-mers in the database called hits. Since BLAST has already pre-processed and indexed the databases for the occurrences of all words in each sequence in the database, this search is fast. If a match is found, then it means that there is a possible alignment between the query and database sequences.

3. Each word hit is extended in both directions without introducing gaps, and we try to extend in both directions so as to increase its score. When we reach a region where extending seems only to lower the segments score, we abandon the search. Each resulting gap-free alignment is called an HSP. When each hit is extended, we are determining if an MSP that includes the w-mer scores greater that S, the preset threshold score for an MSP. Since pair score matrices typically include negative values, extension of the initial w-mer hit may increase or decrease the score. Accordingly, a parameter X defines how great an extension will be tried in an attempt to raise the score above S.







As we can see, BLAST is a pair wise local alignment search tool that is designed to operate more quickly than exact methods, but without a guarantee of finding the best possible alignment. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. As mentioned above, BLAST uses a heuristic algorithm, which seeks local as opposed to global alignments and is therefore able to detect relationships among sequences that share only isolated regions of similarity [1]. The other thing BLAST provides is the statistics of the sequence comparison. So rather than just ranking things by score and more or less leaving it up to the user to figure out when a match was significant and when it wasn't, BLAST could give a solid number indicating which matches were worth looking at. The disadvantage was that the original BLAST didn't allow gaps in the alignments. Thus new versions of BLAST were introduced to allow gap as well as speed up the program.

The original BLAST finds many alignments involving a single database sequence which, when considered together, are statistically significant. If we overlooked any one of these alignments, then we compromise the combined result. By introducing an algorithm for generating gapped alignments, it becomes necessary to find only one ungapped alignment rather than all the ungapped alignments included in a significant result. The new gapped alignment uses dynamic program of Smith-Waterman to extend a central pair of aligned residues in both directions.

Basically the **gapped BLAST** algorithm allows local gaps i.e. deletions and insertions to be introduced into the alignments. Allowing gaps means that similar regions are not broken into several segments. The advantage of this is only one of the component HSPs has to be located for the combined result to be generated successfully. The scoring of these gapped alignments tends to reflect biological relationships more closely and allows easy biological interpretation over longer regions. In gapped BLAST, which is available at [4], only database sequences containing at least two nearby segment with some similarity to the query sequence are considered for further analysis and extension. This 'two-hit' approach increases the sensitivity of the search as well as its speed, provided there is sufficient similarity between the sequences.

The 'two-hit' method requires the existence of two non-overlapping word pairs on the same diagonal, and within a distance A of one another, before an extension is invoked [5]. To achieve comparable sensitivity, the threshold parameter T must be lowered, yielding more hits than previously. However, because only a small fraction of these hits are extended, the average amount of computation required decreases. This also makes the program three times faster.

Another method of finding related sequences in databases that is more sensitive than pair-wise comparison is to use a profile search. Instead of using a single amino acid at a given position in the query sequence, it is better to use a combination of amino acids known to be present at the same position in that protein and related ones. The search of sequence databases will thereby be expanded to include additional related sequences that might otherwise be missed. The major difficulty with such an expanded search is that an alignment of related sequences must already be available in order to know the variations at each position in the query sequence. A new version of BLAST called **position-specific-iterated blast or PSI-BLAST**

has been designed to provide information on this variation starting with a BLAST search by a single query sequence.

The method used by PSI-BLAST involves a series of repeated iterations and uses a position specific score matrix in place of a query sequence and associated substitution matrix. PSI-BLAST will search a protein sequence database with a query sequence profile, a matrix with rows representing sequence positions and columns representing variations in that position. The motif or profile keeps track of the available variations in the alignment of a set of related proteins. First, the search of the desired protein sequence database is performed using a query sequence. Second, the results of the search are presented and can be assessed visually to see if any database sequences that are significantly related to the query sequence are present. Third, if there are sequences from the database those match the query another iteration can be performed.

The high scoring sequence matches found in the first step are aligned and from the alignment a sequence profile, which indicates the variations at each aligned position, is produced. The database is then searched with this profile. The search has thus been expanded to include sequences that match the variations found in the profile at each sequence position. The results are again displayed, indicating any newly discovered sequences that are significantly related to the profile sequences in addition to those found in the previous iteration. Again another iteration of the program can be performed, but this time including any newly gathered sequences to refine the profile. In this fashion, a new family of sequences that are significantly similar to the original query sequence can be found.

When comparing the original BLAST with its newly extended versions, the gapped version is faster than the original and also allows gaps, i.e. insertions and deletions, which produce local gapped alignment. The gapped BLAST program runs three times faster than the original BLAST. The PSI-BLAST can greatly increase sensitivity to weak but biologically relevant sequence relationships. PSI-BLAST's performance as compared to the gapped BLAST is not bad; it may require little more time per iteration than the gapped BLAST and also has the option of running automatically.

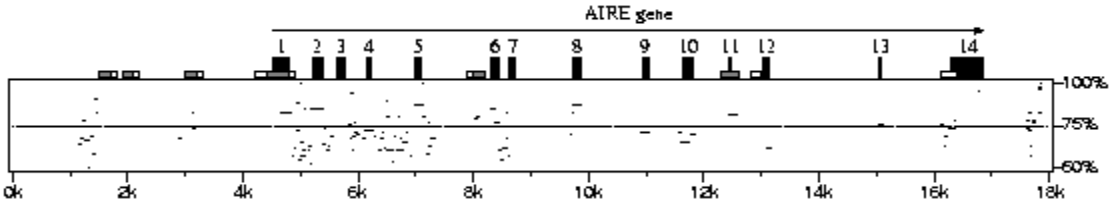
There exists another tool, a web based, for computing alignments of similar regions in two DNA sequences. Previously we saw that BLAST too performed the above, that is comparing two sequences and computing their alignments, but the output of BLAST is mainly in textual form. We also know that BLAST can compare amino-acid sequences thousands of letters long but PipMaker can handle sequences millions of letters long. It can compare the complete genomes of the human and the mouse (Miller).

4.0 PipMaker: A Web Server for Aligning Two Genomic DNA Sequences:

PipMaker is a World Wide Web site that also produces informative, high-resolution displays of the resulting alignment. PipMaker is a visualization tool offering different displays, such as a percent identity plot or pip for short. The pip tool shows both the position in one sequence and the degree of similarity for each aligning segment between the two sequences in a compact and easily understandable form. The other display is a plot of locations of those segments in both species. PipMaker generates graphical output as a PDF document by default, or optionally as a PostScript document

The PipMaker program allows long sequence files, containing as many as millions of nucleotides to be analyzed. Bacterial genomes can be up to 6million nucleotides, as can the regions of conserved syntenic loci in human and mouse chromosomes [2]. When dealing with sequences that are very long, the alignment program must be very fast, naturally. To achieve speed the series of Blast programs have been used. Since the input is very large, the output containing all local alignments between two long sequences will also be lengthy. Therefore the program must present the large volume of output to the user in a compact, understandable form, which is pip as mentioned above.

The PipMaker outputs the position in one sequence of each aligning gap-free segment and plots its percent identity. As a complementary display, it also provides a plot of the position of each aligning segment in both species. We refer to these as dot plots, even though matches shown in conventional dot plots need not be contained within a statistically significant alignment and those in our plots are. Both displays allow rich annotation to be plotted along the appropriate axes to aid in correlating aligning segments with functional or structural features of the sequence.



To generate a pip, such as the one shown above, PipMaker requires four user-supplied files. The first sequence file is depicted along the horizontal axis. Scattered repeats in the first sequence are indicated by various kinds of triangles, whose locations are supplied by a mask file of the first sequence. (The user generates this file using the RepeatMasker program.) A file of gene and exon positions allows PipMaker to draw the locations of exons and indicate the directionality of genes, shown as black boxes and long arrows, respectively. Finally, the user provides a second sequence file. CpG islands in the first sequence are independently determined by PipMaker and are shown as low boxes.

PipMaker compares the first and second sequences. Alignments are plotted according to the position in the first sequence file. The light horizontal line through the middle of the plot indicates 75% nucleotide identity. This version of PipMaker compares the first sequence with both the second sequence and its reverse complement, so matching regions need not occur in the same orientations and relative positions in the two sequences [1].

5.0 A Workbench for Multiple Alignment Construction and Analysis (MACAW):

Multiple sequence alignment is a good tool for examining molecular evolution, both in the form of the DNA and in the form protein. While a pattern found in only two sequences may not appear significant, the same pattern found in many sequences may be very interesting. But researchers can't relay only on the brute force of their algorithms and computers. They must interact freely with their tool to obtain good and useful results. That was the main reason for developing MACAW tool, tool for locating, analyzing and editing of the multiple alignments. It is not just one algorithm, but it is a compound of pure mathematical and statistical algorithms and empirical approaches to the problem of alignments. It was developed ten years ago, when Microsoft Windows became strong enough, and when they introduce new, more natural, way of interaction with computer.

In short, we can say that MACAW incorporates several features:

1. Regions of local similarity are located by the search algorithm that avoids many of the limitations of individual algorithms
2. The statistical significance of blocks of similarity is evaluated using a mathematical theory developed during paste 15 years;
    1. User can edit each block by moving its boundaries or by eliminating particular segments, and blocks may be linked to form a composite multiple alignment.

The need for this kind of program grew because multiple alignment is not so straightforward problem. It can't be easily defined, and for solving this problem several approaches must be used. Life of a scientists would be nice, but there is one problem. For a set of n protein sequences each of length l, a region of similarity common to all may begin anywhere in each sequence and it should be compared to all other sequences, that is l^n alignments should be checked. This search space is too large. For l of the order of 100 and n of the order of 3 this task is impractical. To solve this problem MACAW imposes a single condition on the alignment it seeks: that all segments show a minimal amount of mutual similarity. This can examine all of the search space in $O(n^2 l^2)$ time. The remaining alignments can then be carefully analyzed and their statistical significance assessed.

Terminology

Every new science introduces new terms in use. The same thing is with bioinformatics. Some twenty, or twenty five years ago computer scientist and engineers introduced new term in common language. Now, people from bioinformatic and genomic community are finding new usage for old terms.

- For a set of ***n*** sequences, one subset of segments of some specific length from each of ***m*** (***m*<*n***) sequences forms a ***m-block***, or simply a ***block***.
- Any set of m sequences locked into a specific alignment, with no gaps allowed, is a ***m-diagonal*** or simply a ***diagonal***.
- An aligned set of n amino acids is ***n-column*** or a ***column***.

Algorithm

This tool consists of a few different tools, but central part is search routine which identifies diagonals that may contain a block of interest, but everything starts from comparison of two sequences. Few methods can be used to compare sequences. When two protein sequences are compared set of scores is used to calculate their difference. Whether gaps are allowed, or not, set of is used, and this tool uses PAM-250. The score of the block is the sum of the scores assigned to each of its columns. Score of the column is the sum of all pairwise similarity scores of the amino acids it comprises. Those **SP** scores are called "Sum of the Pairs". MACAW can use some different, more biologically realistic set of scores. Search routine of MACAW seeks only blocks in which all pairs of segments are contained in pairwise subalignments with score greater than or equal to some threshold ***T***. This threshold should be chosen so that about 10% of the diagonals are marked. For ***n*** sequences of total (aggregate) length ***L***, each sequence must be compared with others, and this takes O(L^2) time. This is the most time-consuming step.

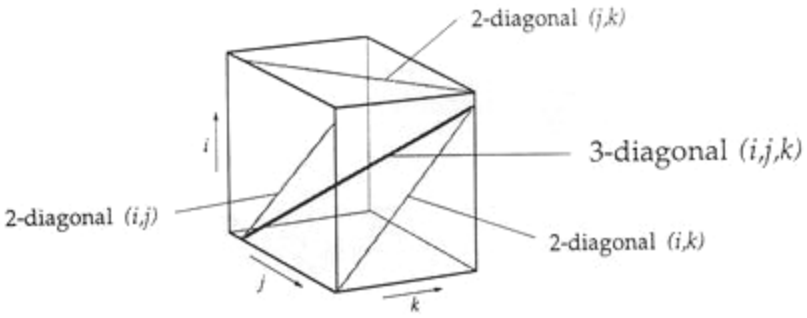


Fig 1.

For any multiple diagonal, all implied 2-diagonals must have been marked during first phase. Thus a diagonal containing three sequences is formed only when each of the three 2-diagonals it contains has been marked. A diagonal is reported only if there is no way in which to extend it further.

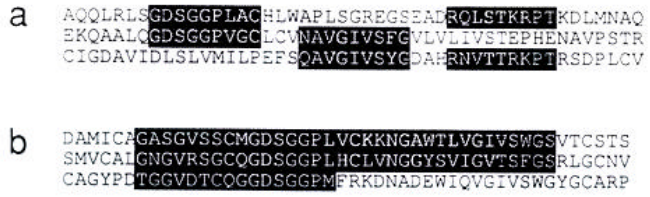


Fig 2. (a) Pairwise similarities in a 3-diagonal that have no mutual agreement. (b) Pairwise similarities in a 3-diagonal that agree but vary in extent.

The basic problem is that homologies that exist in a n-diagonal may be represented by n-blocks, or by 4, 3 or 2 blocks or some other combinations of disconnected blocks. To solve this problem MACAW uses heuristic approach in two steps:

1. Program searches for the highest 2-block in every 2-diagonal using threshold ***T*** and marks the amino acid pairs it contains.

2. Columns are represented as graphs. Every vertex is amino acid and edge between them exist if they are marked in step 1. MACAW searches for the most connected (with as many as possible edges) graphs and connects them into one block.

This procedure doesn't mark all the relations between sequences, so MACAW allows user to edit blocks. To help user every amino acid is colored according to the number of edges it is connected.

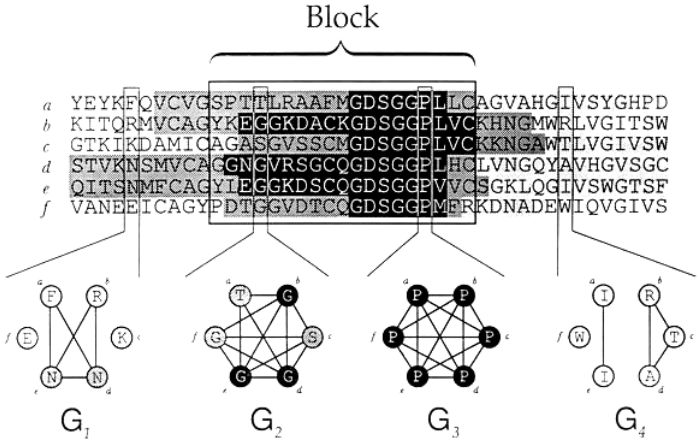


Fig 3. Parsing a diagonal.

Each column is represented by a graph showing a putative relationship among the residues. Adjacent columns whose graphs connect residues from the same sequences are coalesced into blocks. Each residue is colored to indicate the number of edges it touches in the graph representing its column.

When number of input sequences n grows, the number of reported diagonals do not explode. For an arbitrary diagonal of m sequences every of its $\binom{m}{2}$ 2 -diagonals has 0.1 probability of being reported. Whole diagonal has probability $(0.1)^{\binom{m}{2}}$

If each sequence has length l, the number of m-diagonals is approximately $m\binom{n}{m}l^{m-1}$

While this number grows exponentially with m, first probability decreases with m^2. Threshold T may be upward adjusted to decrease total probability that random 2-diagonal is marked.

Statistical significance

The statistical (and biological) significance of a block depends on a model of chance. Authors of the MACAW assume that at any position each type of amino acids has a probability of occurring and that those probabilities are position independent and have no Markov dependence.

Statistical results give formula for SP score for n-block with score S, found by searching sequences of length $l^1, \ldots l^n$:

$$1 - \exp(-KNe^{-lS})$$

N is the product of the l^i, and K and l are constants determined by the possible scores for an n-column and their corresponding probabilities.

Examples

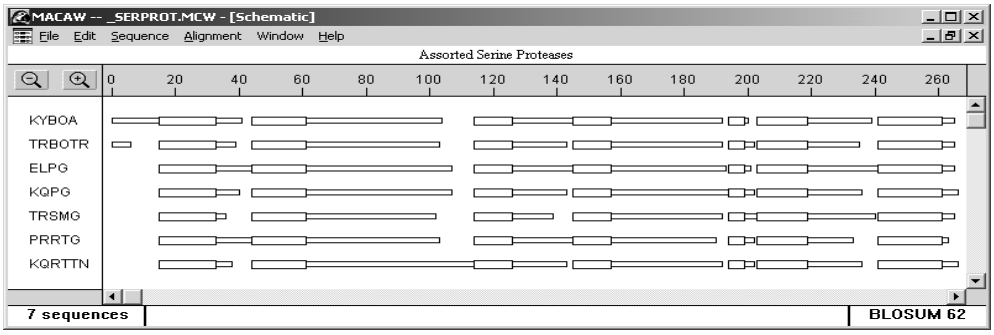


Fig 4. Schematic view of sequences

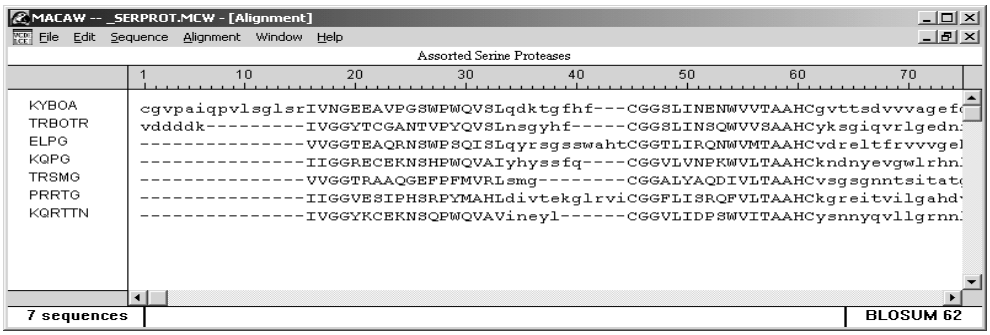


Fig 5. Alignment

6.0 Motif Explorer-A tool For Interactive Exploration of amino acid sequence motifs

Short amino acid sequence patterns, called motifs, are conserved sequence fragments or set of fragments with biological significance. They play an important role in molecular biology research. In order to determine the sequence containing a motif, its description is matched against a database of known sequences. While a number if tools for locating motif sequence databases have been developed, no existing tool performs fast enough to allow interactive searching of the entire databases. Interactive searching enables biologists to explore the effect of changes to a motif by analyzing the set of sequences matched by that motif. This process is also known as motif exploration.

Motif explorer is a software tool, which uses indexing structure based on generalized suffix trees for high performance motif searches, and makes the interactive motif exploration possible. It's fast enough to provide answers within seconds.Regular expressions are a powerful and efficient method for describing patterns in strings, both in text and in biological sequences. Algorithms for regular expression based pattern searching fall into two broad categories: direct searching and indirect searching.

These two categories are sometime also referred to as searching without preprocessing and searching with preprocessing, respectively. A direct searching uses an algorithm scanning along the sequence being searched to detect the presence of a pattern. The detection task is usually accomplished by a variant of a state machine generated based on the pattern being searched and driven by sequences being scanned. The indirect searching preprocesses the sequences from the database to be searched and stores the resulting information in an indexing structure which is used to speed up subsequent searches. Thus an indirect search is performed in the feature domain encoded within the indexing structure as opposed to the sequence domain, as in the case of direct search. The performance of the indirect search algorithm is

dependent on the type of indexing structure used. One of the indexing structures that can be used for indirect pattern searching in sequences is the generalized suffix tree.

A generalized suffix tree, or GST, is based on several simple structures. Like the trie and Patricia tree. A trie is an indexing structure used for indexing sets of key values of varying sizes. A trie is a tree in which the branching at any level is determined by a partial key value. A Patricia tree is a trie in which information in each node is used to indicate which portion of the key is used to determine branching. Next, a suffix tree is a Patricia tree built over the set of all suffixes of a given sequence. A collection of more than one suffix tree in one representation (suffix tree) is called the generalized suffix tree.

Given a Regular Expression (RE), first a deterministic finite automata (DFA) is constructed. The RE search algorithm operates by traversing the GST depth-first from the root along edges matching state transition labels in the DFA. A substring matching the RE is located when a transition to an accepting state in the DFA occurs.

GST-based regular expression search performance prediction:

A regular expression recognizes a set of strings. By constructing a trie of all strings recognized by a given RE a trie representation of an RE can be obtained, such trie are referred as RE-trie. Expressing REs in terms of RE-tries makes the RE search algorithm, more intuitive and allows for its performance to be analyzed in terms of GST matching. The time necessary to perform the search is proportional to the size of the portion of the RE-trie that matches the GST being searched. The size of the matching portion of the RE-trie is bounded by the size of the entire RE-trie. The only way to exactly determine the portion of the RE-trie that is going to match a given GST is to perform the actual match. An estimate if the matching portion can, however, be obtained by analyzing the structure if the RE-trie and the structure of the GST of the database. The computation has to be performed for the entire RE-trie. However, it can also be performed directly on the DFA by replacing RE-trie branches with DFA state transitions. The computations need not have to be completed to obtain an accurate estimate. For infinite RE-tries the computations cannot be completed. It is sufficient to traverse the RE-trie down to a depth at which probability becomes negligibly small, thus does not significantly contributing to the estimate.

Implementation and maintaining GST-based indexing structures for large sequence databases requires intricate implementation and significant computational resources. The key performance obstacle is the need to update large data structures residing in disks, which are many orders of magnitude slower than main memory. The technique developed is to allow GST to be created and processed as linear steams of data, using limited main memory and without random access to the entire structure.

The Motif explorer is an interactive sequence motif exploration tool using GST structure to provide performance levels acceptable for interactive searching. It combines the implementation of large-scale GST structures with the GST regular expression search algorithm. The motif explorer uses the GST regular expression search performance predictor to improve the performance of the search. It is equipped with a World Wide Web (WWW) based interface. And so is easily accessible through the internet. Its accessible at http://lenti.med.umn.edu/gst/MotifExplorer.html

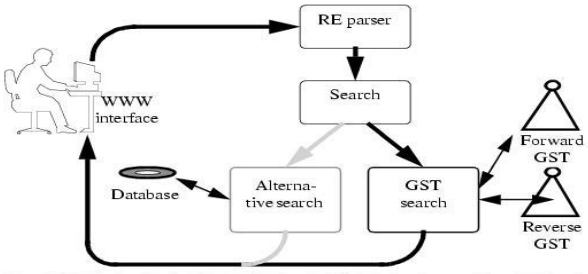


Figure: Main Functional block of the Motif Explorer Tool.

Motif Explorer accepts queries in the UNIX-style regular expression format as well as the ProSite database format. After parsing, the query is passed on to a search time prediction module. Based on the result of the prediction the forward or reverse GST structure is chosen for searching. The predictor allows the best strategy to be selected for a given regular expression. If only one search structure is available, which may happen in case of limited disk resource, it is used. Optionally, if the prediction algorithm indicates a very long search, the user may be warned that an answer cannot be quickly produced or an alternative, fallback search algorithm may be used. The search results are then formatted and presented to the user in the hyper-text document format. The search result include the matching sequence as well as additional information obtained from the GST structure, such as unique matching sequence fragments and the frequency of their occurrence within the database. Motif Explorer also includes WWW links to other appropriate databases containing information of the matching sequences.

Occasionally, high-redundancy patterns can cause the Motif Explorer's performance to degrade below that of a linearly scanning algorithm. For it a hybrid system is constructed, which combines the GST-based system with the linear scanning system. And based on the GST regular expression search performance predictor result values, appropriate algorithm is chosen.

The motif Explorer is less by approximately an order of 20 when compared to the grep program in its average and median search timings.

7.0 Five experimental methods for finding conserved sequences in multiple alignments of gene regulatory regions:

Bioinformational problems are new rapidly growing field of science and new approaches are developed to meet the requirements of the biology, chemistry and computer architecture. Special task is finding conserved sequences in multiple alignments in genes of different species. A conserved character in DNA is one that was probably present in the common ancestral species and has been preserved in the contemporary species being examined. Those conserved regions can help us find our animal roots and use them to find new drugs. For a given set of conserved sequences, the main problem for molecular biologists and computer scientists is distinguishing functional regions from those whose similarity reflects the residual common ancestral sequence that has not yet changed via evolutionary drift. Scientists and computer programmers all around the world are trying to develop or improve computer algorithms for finding highly conserved block within previously computed multiple alignments, primarily for DNA sequences.

Here will be presented five of them. Two of the methods are already in common use; these are based on good column agreement and high information content. Three additional experimental methods find blocks with minimal evolutionary, blocks that differ in at most **k** positions pre row from a center sequence that is unknown a priori. The center sequence in the latter two methods is a way to model potential binding sites for known or unknown proteins in DNA sequences

A multiple alignment generates a matrix with each DNA sequence occupying a row so that each nucleotide is placed in an appropriate column. A conservative group of columns, or block, can be identified as conserved based on a number of approaches. The simplest is to compute the level of similarity in each column and find blocks that fit user-defined criteria for the degree of similarity per column and the length of the block. This column agreement approach does not take into account the effects of nucleotide frequency in the genes under consideration and thus is developed a metric called information content that incorporates both nucleotide similarities and overall nucleotide composition as a measure of column similarity.

Two of five methods described in this part are aimed at finding protein-binding sites on DNA. Such sites are usually a series of consecutive positions, one or more of which can vary somewhat without measurably changing the binding affinity. Thus it is desirable to examine a series of neigbouring positions in each row when finding blocks. Each row-based method allows up to **k** mismatches per row; in one method the mismatches are relative to a specified 'center' sequence (e.g. the human sequence) and in the other the mismatches are relative to an unknown 'center' sequence.

Certain parameters are common to all of the tools/programs/algorithms. The minimum length of the regions to be reported and the minimum number of sequences, which must be active, are selectable by the user. The search can be conducted in the entire alignment or it can be restricted to a portion specified by a given range in any of the sequences.

Five tools:

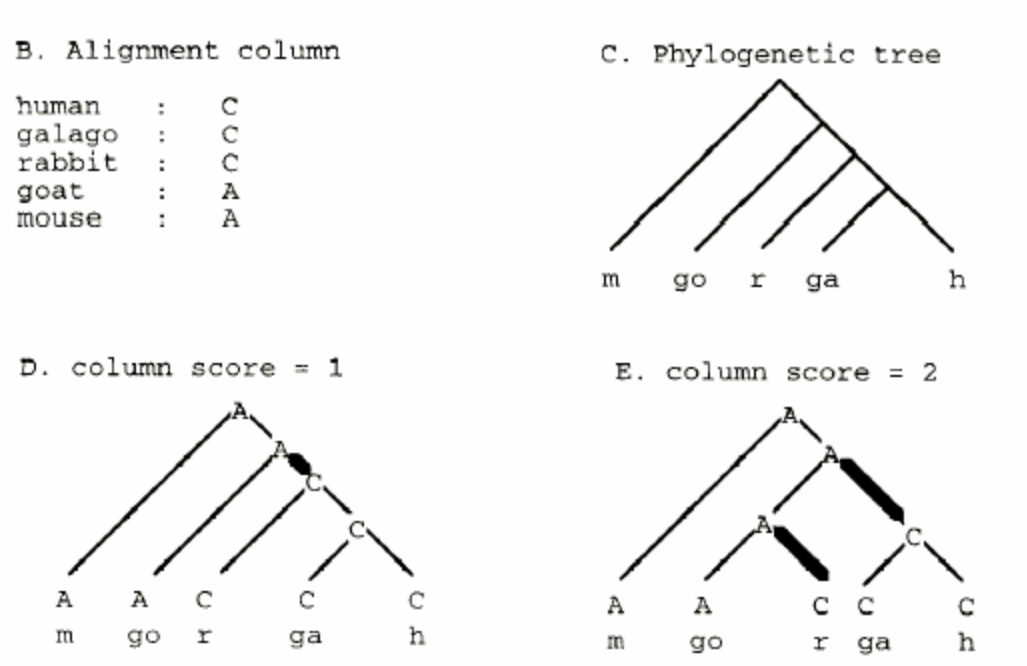
agree : This utility locates regions in a given alignment that have good column agreement.The columns are examined individually to determine whether or not they meet a user-specified threshold for letter agreement, and runs of columns passing this test are reported.

infocon : The length of the region is often a reliable indicator that some functionality was preserved across the species, but conservation doesn't need to be perfect and such regions might be fragmented into conserved pieces too small to be detected, so a systematic way to link the smaller regions is needed. The two utilities **infocon** and **phylogen** are trying to solve this problem. The idea is to assign a numerical score to each column and then look for runs of columns meeting the following two conditions:

2. their cumulative score (obtained by adding together the individual column scores) is no smaller than the score of any of their sub-runs;
3. they are maximal with this property, i.e. they are not contained in any longer run having the property 1.

The **infocon** tool finds full runs of columns with high information content in the given alignment. To do this, each column is assigned an intermediate score that measures its information content, based on the frequencies of the letter both within the column and within the alignment as whole. The exact value of this score is the $1/L$ multiplied by the logarithm of the likelihood ratio obtained for the frequency of letters within the alignment and within the column under examination, where L is the number of active sequences in the alignment column.

phylogen : This program scores the columns by their evolutionary relationships among the sequences of the given alignment implied by a supplied phylogenetic tree. The phylogenetic tree has a leaf node for each species and each internal node represents a putative common ancestor for the species in its sub-tree.



For each column, **phylogen** assigns to each leaf node the letter from the alignment row of the corresponding species, and labels the internal nodes so as to minimize the total number of changes in the tree. This number is the initial score associated with the column and it is computed as the total edge weight of the labeled tree, where an edge has weight 1 if it corresponds to a letter change, and 0 if it connects two nodes labeled with the same character. The root label is named the **ancestral** for the column. Well conserved columns will have low scores, but the selection algorithm is geared toward maximization, the column scores are adjusted by subtraction them from a suitable **anchor value**. This value must be chosen carefully because both the negative and positive values may occur. It can be calculated by the program as

either the current number of active rows for a column or the current number of active rows not containing a gap, or it can be set as some non-negative number.

kkno : This program scans the alignment to determine, starting at each position, the longest region in which no row differs from a specified, known 'center' sequence in more than **k** positions.

```
center: A C C G T G C A G
        1 2 3 4 5 6 7 8 9
human : A G C G T G C A C
rabbit: A C C G T A C A T
mouse : T C C G T A C A C
```

kunk : This program is similar to **kkno** except that the center sequence is not known a priori; instead, the program computes the 'best' center sequence for each conserved region it finds. This center sequence can be thought of as belonging to a common ancestor of the species represented in the alignment or as a potential binding site for known or unidentified proteins.

For each column in the alignment, the algorithm recursively examines all possible center sequences starting at that position to see how far the region can be extended and back –tracks when the extension becomes impossible. The quality measure for assessing a potential center sequence is the sum of the squares of the number of mismatches between it and the alignment sequences within the region. A lower value indicates a better candidate for the center sequence. Only characters within a column can be used in the center sequence.

This method is more flexible than **kkno** in that it allows consecutive letters in the center sequence to be drawn from possibly different alignment rows. It allows the letter inhabiting a certain position in the center sequence to vary between applications of the procedure for different starting columns.

Parameter calibration for each method

Larger percentage threshold for column similarity in **agree** or a lower number of permitted mismatches in **kunk** and **kkno** lead to a smaller number and shorter length of the reported regions. For a fixed required minimum region length, regions obtained by **phylogen** with a larger anchor value always include those obtained with smaller ones. Regions produced by **infocon** decrease in number and extent as the value of the score adjustment parameter increase.

Comparison of five methods

There is no superior method to the others and in fact, their similarity is the proof that they are strong enough. The goals of the user will choose which of the methods is going to be used. Highly conserved motifs can be detected by each of the methods but with slightly different end-points.

Good results obtained by agree, infocon and phylogen require calibration against the data set of known functional regions, because it is very difficult to obtain good values before the experiment is started. Methods **infocon** and **phylogen** give best results, but it is difficult to predict optimal anchor value.

The approach of first making the alignment and then searching for conserved motifs has some limitations, but gives relatively good results. Quality of the methods depends on the quality of the alignment, and sometimes obtaining a good multiply alignment can't achieved.

8.0 Conclusion

The area of bioinformatics is relatively young, but it is maturing very fast because the problem it deals with is growing exponentially, and not only because the amount of data is bigger every day, but because new methods and ideas are being developed and introduced in life science. Good biologist or biochemist must use computer programs if he wants to achieve any significant result. There is task for computer engineers and scientists and that task is developing of the new tools and methods to solve biological problems. The main problem is that you can't define biological problems very easy, so computational solution is not straightforward. Some empirical approaches must be used, and heuristics are involved in solutions. This paper presented few new tools based on the straight mathematical theory but reinforced by the empirical findings heuristic approaches. They use properties of modern operating systems and properties of Internet to reach their goal.

9.0 References

[1] Altschul, S. F., W. Gish, E. Myers, W. Miller and D. J. Lipman (1990) A basic local alignment search tool. J.Mol. Biol. 215, 403410.

[2] Alexander Pertsemlidis, John Fondon III (2001) Genome Biology, 2(10): reviews 2002.1-2002.10.

[3] Karlin, S. & Altschul, S.F. (1990) "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes." Proc. Natl. Acad. Sci. USA 87:2264-2268.

[4] NCBI, http://www.ncbi.nlm.nih.gov/

[5] Altschul, S. F., T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. J. Lipman (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research 23, 33893402.

[6] Micheal S. Waterman. Introduction to computational biology: Maps, sequences and genomes. Chapman & Hall, 2000.

[7] W. R. Pearson, *Rapid and sensitive sequence comparisons with FASTP and FASTA*, Methods in Enzymology Vol. 183 Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences (R. F. Doolittle, editor), pp. 63-69, 1990.

[8] W. R. Pearson and D. J. Lipman (1988), "*Improved Tools for Biological Sequence Analysis*", PNAS 85:2444- 2448,

[9] W.j.Wilbur and D.J. Lipman, Proc. Natl. Acad. Sci. USA 80, 726 (1983).

[10] G. D. Schuler, S. F. Altschul and D. J. Lipman, A *workbench for multiple alignment construction and analysis*, Proteins: Structure, Function and Genetics, Vol 9, 1991, pp. 180-190.

[11] P. Bieganski, J. Riedl, J. V. Carlis and E. R. Retzel, *Motif explorer—a tool for interactive exploration of amino acid sequence motifs*, proceedings of the first Pacific symposium on Biocomputing, 1996, pp. 705-706 (IEEE Computer Society Press).

[12] P. Bieganski, J. Riedl, J. V. Carlis and E. R. Retzel, *Generalized Suffix Trees for Biological Sequence Data: Application and Implementation*, Proceedings of the 27th Hawaii International Conference on System Science, Maui, January 4-7, 1994.

[13] N. Stojanovic, L. Florea, C. Riemer, D. Gumucio, J. Slightom, M. Goodman, W. Miller and R. Hardison, *Comparison of Five Methods for Finding Conserved Sequences in Multiple Alignments of Gene Regulatory Regions*, Nucleic Acids Research, Vol 27, 1999, pp. 3899-3910.

[14] S. Schwartz, Z. Zhang, K. A.. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison and W. Miller, *PipMaker: A Web Server for Aligning Two Genomic DNA Sequences*, Genome Research, Vol. 10, Issue 4, April 2000, pp. 577-586.

[15] http://bio.cse.psu.edu/pipmaker/pip-instr.html

[16] http:// bio.cse.psu.edu/pipmaker/papers/genome-research/

[17] ftp://ftp.ncbi.nlm.nih.gov/pub/schuler/macaw/

[18] http://www.bris.ac.uk/pathandmicro/services/CGR/Common%20Files/MACAWinstructions.htm