2002

RNA structure energies applied to genome sequence search

Davinderpal S. Wander
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Wander, Davinderpal S., "RNA structure energies applied to genome sequence search" (2002). *Master's Theses*. 2341.

DOI: <https://doi.org/10.31979/etd.r7r3-zgms>
https://scholarworks.sjsu.edu/etd_theses/2341

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

**RNA STRUCTURE ENERGIES APPLIED TO GENOME
SEQUENCE SEARCH**

**A Thesis Presented to
The faculty of the Department of
Chemical Engineering
San Jose State University**

**In Partial Fulfillment
of the Requirements for the Degree
Master of Science**

**By
Davinderpal S. Wander**

August, 2002

UMI Number: 1410451

Copyright 2002 by
Wander, Davinderpal Singh

All rights reserved.

UMI[®]

UMI Microform 1410451

Copyright 2002 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2002

Davinderpal S. Wander

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT
OF CHEMICAL ENGINEERING

Dr. Melanie McNeil

Dr. Brooke Lustig

Dr. Elaine Collins

APPROVED FOR THE UNIVERSITY

ABSTRACT

RNA STRUCTURE ENERGIES APPLIED TO GENOME SEQUENCE SEARCH

By Davinderpal S. Wander

This research was mainly concerned with the feasibility of the possible use of RNA structure energies in genome sequence search. The primary objective of this research was to determine if there exists a correlation between the energy required to anneal two nucleotide strands and the BLAST score associated with these strands. A total of 153 (back-translated) proteins and RNA sequences were used. A BLAST search of GenBank was performed for each of the 153 sequences and all resulting alignments were used to compute the corresponding energy value. The total resulting alignments were 38,995.

Results reveal that there is a statistically significant linear correlation between BLAST scores and the corresponding RNA secondary structure energies. The average R-value was determined to be 0.91 and the boundary statistical probability of exceeding R was determined to be 0.001. In addition, the energy rules appear to be more sensitive and generalized than the BLAST scoring scheme.

ACKNOWLEDGEMENTS

I would like to thank all my reading committee members for their support and guidance throughout this research. I express my appreciation to Dr. Melanie McNeil for making this research possible. Further, I like to especially thank Dr. Brooke Lustig for his continued support, critical evaluation, and guidance of this work. I also thank Dr. Elaine Collins for providing her evaluations and suggestions. Finally, I thank my parents for their general support.

TABLE OF CONTENTS

Chapter	Page No.
1. Introduction	1
2. Literature Review	8
3. Research Objectives	22
4. Methods	23
5. Results and Discussion	29
6. Conclusions	55
References	56
Appendices	
A. Appendix-I (List of 153 PBD Identifiers)	58
B. Appendix-II (Correlation Plots of all 153 Protein & RNA)	59
C. Appendix III (Human and E. coli Source Proteins)	110
D. Appendix IV (Perl and Java Codes Used)	111

LIST OF FIGURES

1. Dot-plot matrix of spider monkey and chimpanzee partial DNA	3
2. Global alignment versus local alignment	4
3. Smith-Waterman alignment for two arbitrary sequences	5
4. Hydrogen bonding and Base stacking phenomena	15
5. Bulge, hairpin, and internal loops.	18
6. Correlation plot for back-translated Modified α_1 -Antitrypsin	31
7. Plots of lowest, average, and highest R^2 value	33
8. Aggregate correlation plot of all 38,995 alignments	35
9. Aggregate correlation plot excluding 1c2w	36
10. Aggregate correlation plots of low and high scoring regions	37
11. Aggregate correlation plots for results from amino acid and RNA sequences	39
12. Correlation plots classified by two different sources	48
13. Comparison of BLAST scoring versus the energy rules	50
14. Distribution of duplexes with ΔG of -1 and -3 kcal/mol	53

LIST OF TABLES

1. Thermodynamic parameters for propagation sequences	17
2. Free energy change for loops in kcal/mol	19
3. P-values for the 153 query sequences and for some aggregate plots	45

1. INTRODUCTION

The field where computational methods are used to gain biological insight, especially when applied to nucleotide and amino acid sequences, is known as Bioinformatics. Fredj Tekaia of Institut Pasteur defines bioinformatics as – The mathematical, statistical and computing methods that aim to solve biological problems using DNA and amino acid sequences and related information. Bioinformatics is, therefore, inherently a multidisciplinary subject as is obvious in the development of human genome project, a vast bioinformatics-related project.

Several genome projects completed hitherto and in progress have resulted in gathering the nucleotide sequences of many species. Recent advances in Bioinformatics have accelerated the rate in obtaining these sequences. These DNA sequences (and amino acid sequences in case of proteins) are primarily made available through databases such as National Center for Biotechnology Information (NCBI), GenBank (a National Institutes of Health database), and Protein Data Bank (PDB).

Once the genome sequence of an organism is determined, the next step is often to compare it with other genome sequences, study phylogenic relations among the sequences, and identify functional groups. Differences and similarities between the two organisms' nucleotide sequences provide useful information on their evolutionary development.

Advances in DNA sequencing such as automated DNA sequencing techniques have made it easier to obtain such information. The databases used to store this information provide actual sequences of an organism. A well-aligned sequence is a

subject sequence where the residues of the DNA query and subject sequences are the same. A well-aligned sequence provides useful information in terms of evolutionary changes that take place between ancient and modern organisms and allow for functional studies (where the function of specific sequences is studied, in proteins for example). In cases where sequences show similarity and good alignment, it may be concluded that the query sequence might have the same function as the subject sequence. In addition, advances in this area of biotechnology have provided insight into finding cures for diseases as well. This is genetically done by manipulating the nucleotide sequences and therefore protein products.

In searching databases for best alignment of a query nucleotide sequence of interest, the general search algorithms are Dot Matrix, Smith-Waterman algorithm, FastA (Fast Alignment) algorithm, and BLAST (Basic Local Alignment Search Tool) algorithm. Much time and money may be saved in laboratory experimentation by making good use of these algorithms. The proliferation of nucleic acid and amino acid sequences now available in databases, however, demand the need for study and further development of these algorithms for sequence alignment.

Each algorithm uses a different method to evaluate alignments. In using the Dot Matrix, two sequences are compared by placing a dot in the I^{th} row and J^{th} column of the matrix if the I^{th} nucleic acid residue (or amino acid) is identical or similar to J^{th} residue (or amino acid). In this manner a diagonal line composed of the dots indicates a match of the nucleotide sequences, and gaps represent mismatch in sequence or a mutation in the sequence. One of the major advantages of using a Dot Matrix is that it

shows *all* possible matches of residues between the two sequences. Figure 1 below illustrates this idea with spider monkey and chimpanzee hemoglobin intergenic DNA, note the gaps (or mutations) represented by misplacement of the center of diagonal line.

Figure 1. Dot-plot matrix of spider monkey and chimpanzee partial DNA
(Source: <http://helix.biology.mcmaster.ca>)

A Smith-Waterman table structurally appears to be somewhat similar, where a two dimensional table is constructed with one sequence on the vertical axis and the other on the horizontal axis. Each cell in the table represents the alignment/misalignment of the corresponding residues. A scoring scheme is used to determine the best alignment. A match in the residue sequence is given +1 points, a

mismatch is assigned $-1/3$ points (as penalty) and an open gap scores $-1+1/3k$ where k represents the extent of gap. The following equation is used to fill in the table.

$$X_{i,j} = \max\{X_{i-1,j-1} + \text{Score}(a_i, b_j), X_{i-1,j} - \text{Penalty}(k), X_{i,j-1} - \text{Penalty}(k), 0\}$$

Using this scheme the table is filled in and the highest score in the table is then backtracked until the first zero is reached to obtain the local alignment (1). X_{ij} above is the score of the alignment of sequence a (ending with residue i) and sequence b (ending with residue j). Figure 2 below shows the difference between local and global alignment. Most of the database search algorithms provide local alignment as opposed to global alignment. Figure 3 on the next page shows a Smith-Waterman alignment table for two arbitrarily chosen sequences.

Figure 2. Global alignment versus local alignment

In addition to the above difference, in global alignment the gaps at the end of the sequence are penalized just as internal gaps but in local alignment the gaps at the end of the sequence have a penalty of zero.

Figure 3. Smith-Waterman alignment for two arbitrary sequences
(Source: www.maths.tcd.ie)

Smith-Waterman is also considered to be the most sensitive (and computationally intensive) algorithm. Sensitivity is defined as the ability to identify similarity between sequences and selectivity on the other hand is defined as the ability to identify false alignments between sequences.

The FastA algorithm takes advantage of both the Dot Matrix approach and the Smith-Waterman approach. Whenever there is a match between the two sequences being compared, a dot is placed in the matrix just as is done in the Dot Matrix approach. Thus a diagonal line represents a match in sequences. After this initial

alignment, “restricted” Smith-Waterman alignment is performed on this region (2). Just as FastA, BLAST is another heuristic algorithm. As a screening step, BLAST limits the number of matches by using a “matrix-defined threshold” for the current specified matrix of interest (3). Sequences scoring above the matrix-defined threshold are retained and used for alignment.

The present day sequence alignment algorithms mostly function by taking the maximum score which is calculated by adding all the “weights” associated to matches and subtracting a “penalty” for each gap during the alignment. The Needleman-Wunsch algorithm was the first to use this principle for sequence alignment. The key point to note here is that most of the present day alignment strategies simply involve comparing the query nucleotide sequence with subject sequences to find the best match and alignment. One of the major drawbacks of these database search algorithms is that they are based on arbitrary user defined parameters (such as window size and gap penalties); therefore different users often end up with different results.

One novel approach to sequence alignment is to use thermodynamic parameters to obtain sequence alignment. The idea behind this approach is that more energy is required to bifurcate highly stable base pairs than the energy required for weakly linked base pairs (4). In case of sequence alignment, these energies represent the hydrogen bond energies and base stacking energies between matched/mismatched base pairs. Therefore, an increase in the number of mismatches between two aligned strands will result in an increase in free energy (ΔG becomes less negative) signifying the base pair stability decrease.

This approach circumvents the problems encountered with user-defined parameters in database search algorithms. The potential of this approach becoming part of the current rule-based sequence search algorithms is significant.

2. LITERATURE REVIEW

In this section, a variety of alignment methods will be considered and the novel idea of using the so-called energy potentials in predicting the best alignment between two nucleotide sequences is introduced.

Global alignment and local alignment are the two types of strategies generally used in aligning sequences. In global alignment the entire sequence is used for alignment and penalties are used for end gaps, therefore the entire length of the sequence under consideration is aligned (5). The advantage of global alignment is that it has an immediate structural interpretation (6). In local alignment, however, only portions of each sequence are used for comparison. End gaps are not considered in local alignment, and segments are chosen for optimization of comparison scores. As a result, gaps are present in sequences where local alignment methodology is used. Database search algorithms generally utilize local alignment; therefore the search is focused on the most conserved (high similarity) regions of the sequence. Figure 2 provides an illustration of global alignment versus local alignment.

There are many database search algorithms. These dynamic programming algorithms are mathematically intensive programs commonly used for sequence alignment. This was first proposed by Needleman and Wunsch for sequence alignment of two sequences (7). A variation of Needleman and Wunsch was later proposed by Sellers (8), where global alignment methodology was incorporated. Smith-Waterman is another variation of Needleman and Wunsch. Smith-Waterman uses local alignment and is considered more rigorous than some of the other strategies (9). Among these

there are two other widely used database searching algorithms, FastA which is basically a word search (2) and BLAST which is based on maximal segment pairs approach (3). Just as multiple sequence alignment algorithms, database search algorithms also produce results in terms of sequence alignment and generate alignment scores that differentiate subject sequences from query sequences (5).

In database searching, the selected parameters do not always provide the best/most accurate alignments, and as mentioned before, a good alignment is necessary in postulating the insertions/deletions and substitutions in the two sequences being considered. Therefore it is important to have some type of knowledge about the characteristics of the query sequence in order to determine the optimum degree of sequence similarity (6). The database searches simply provide an output in terms of subject sequences of most statistical significance in being close to the query sequence. Due to the fact that most database search algorithms use local alignment rather than global alignment, the results are normally accentuating conserved regions within the two sequences under consideration.

Sellers also proposed a quasi-global alignment algorithm, which is a variation of dynamic programming algorithm that takes the longer sequences as a template and aligns a shorter sequence with it (8). Smith-Waterman, however, is considered to be the most effective in determining similar sequences. Smith-Waterman is really nothing more than a recursive mathematical equation that makes calculations in steps with all further calculations dependent on the former steps. Recall that in the Smith-Waterman scoring matrix the highest achieved score is backtracked until a zero is reached and

this provides a local alignment (1). This local alignment strategy used by Smith-Waterman has some benefits, as Nicholas, Dearfield and Ropelewski (5) point out – in order to achieve a high score the best scoring regions do not need to overcome any effects of surrounding regions of low similarity. For a global alignment however, the boundaries of the alignment are defined by the ends of the sequences and the partial scores within the table are allowed to become negative. At the same time, one of the downsides of dynamic programming algorithms is that more memory/computational ability is required in almost direct proportionality to sequence length.

In order to mitigate some of the computational requirements of the database searches, heuristic programs such as FastA and BLAST have been developed. A heuristic program, as described by Nicholas and coworkers, consists of three steps (5). In the first step, the query sequence and the subject sequences comprising the database are checked via Dot Matrix to identify diagonal region(s) that show alignment. The sequences identified in the first step are then aligned and a statistical check is made to determine the alignment significance in the second step. A restricted Smith-Waterman alignment is then done as the third step. FastA is one of these heuristic programs to take advantage of this strategy and is commonly used (along with BLAST) to search databases for sequence alignment. FastA performs a word search of database sequences with respect to those of the query sequence. The Dot Matrix approach is used to construct a diagonal line and identify initial alignment between the two sequences and a region of specific word size is selected. On this selected region a Smith-Waterman alignment is performed next. Although this approach is faster than

the Smith-Waterman method alone, it has some downsides as well. Pearson (10) points out these limitations as following, the first limitation is the word size parameter which is limited to four for proteins and six for nucleic acids and the second restriction is specified by the window size that limits the total sum of insertions or deletions one sequence can collect with respect to the other sequence in the course of evolution. Therefore FastA has its own restrictions as well as advantages over Smith-Waterman.

Nicholas et al. (5) and Altschul (3) provide a good discussion of the maximal segment pair alignment, which is what BLAST is based on. Maximal segment pair alignment (or MSP for short) results in gap absent sequence alignment with matches as well as mismatches present in the alignment. Similar to FastA in some ways, BLAST takes the sequences and divides them into a list of overlapping words. Next, as briefly mentioned in the introduction, the words scoring above the matrix-defined threshold for the current matrix are used for alignment purposes. Altschul (3) also mentions the possibility to allow for gaps in the extension step but the tradeoff in speed of alignment proved unacceptable. Although BLAST is very sensitive in detecting “weak (but biologically significant) sequence similarities” (3) it also has its own limitations. As in FastA, BLAST also has the word size parameter restriction, and between the two sequences a statistically significant un-gapped conserved segment must be present. These limitations are obviated by further improvement of the algorithm by requiring two non-overlapping words on the diagonal to be present before extending the alignment and thus decreasing the number of initial alignments.

Recently introduced, ParAlign is another alignment algorithm that takes advantage of the heuristic approach. ParAlign is more sensitive than some of the current heuristic algorithms, but this advantage is at the cost of speed as BLAST is observed to be much faster than ParAlign (11).

Many interactive programs have been developed to align nucleotide (and amino acid) sequences but are not much different in operation than those described above (12). For determining highly conserved sequences located within established multiple alignments, programs such as infocon, agree, phylogen, kkno, and kunk are used. These programs also use user defined parameters such as number of sequences to be compared, minimum length of sequences to be compared and etc. The results can either be local or global alignments (13).

Multiple sequence alignment of nucleotides (or of the corresponding amino acids) has become a very important tool in molecular biology. Multiple sequence alignment is defined as an alignment of the residues where gaps are present so that each sequence has the same number of residues plus gaps in the alignment. These alignments can be used to determine evolutionary linkage between species and to find conserved regions and regulatory sequences. There is no question of the importance of aligning sequences, but the concern is really on the notion of accurate alignment methods capable of handling enormously large sets of data. There do exist such algorithms that can mathematically provide an optimal alignment but are limited to relatively short sequences (14). In the case of large scale multiple sequence alignment, enormous computational effort is required. This problem is mitigated through the use

of several heuristic methods developed for the specific purpose of alignment. This has led to the evolution of many programs that use progressive, iterative or mixed strategies (15).

The algorithms considered so far provide acceptable alignments for closely related sequences. They all have particular advantages and disadvantages over one another and in general. Most depend on user defined parameters such as word size, window size, and gap penalties (as discussed earlier) and some are more sensitive than others. Most of the programs considered are also not very effective in determining similar regions of sequences that have little overall similarity. In addition, the computational complexity of these algorithms grows very rapidly as greater number of sequences are attempted to be aligned simultaneously (16). Successive pair-wise alignment of groups of sequences instead of simultaneous alignments of sequences could alleviate the problem of growing computational complexity. Nevertheless, any of these programs yield alignments that simply tell us the residue positions of subject sequences relative to the query sequence, and dependent on the parameters used, the results may not always be consistent between different users. Note that the algorithms discussed so far do not make direct use of any chemical knowledge acquired on nucleotides and are purblind in this sense. They provide no insight into the thermodynamic stability of aligned sequences.

Although confined to RNA secondary structure prediction, thermodynamic parameters have commonly been used in the past (4). RNA is most widely known to exist in the form of single strands, but nevertheless RNA can form secondary

structures like hairpin/bulge loops and unpaired or mismatched terminal base pairs by simply folding back on itself (17). The nearest-neighbor thermodynamic model allows for RNA secondary structure prediction. The assumption of this model is that the stability of an RNA structure is only dependent on the identity of the adjacent base pairs (4). Turner and coworkers also point out that the baseline for this model is that the RNA secondary structure is stabilized due to the base stacking and hydrogen bonding interactions. The base stacking energies represent non-covalent non-H-bonding interactions between the residues that are present in a stacking manner along the inner core. The hydrogen bond energy is obviously due to the interaction between hydrogen atoms and oxygen & nitrogen atoms caused by the difference in electronegativity. Figure 4 depicts the hydrogen bonds between the residues and the stacking phenomena.

Figure 4. (a) Hydrogen bond between residues (b) Base staking phenomena
 (Source: www.blc.arizona.edu/molecular_graphics)

Studies conducted in RNA secondary structure prediction (4,18) provide the thermodynamic parameters for determining oligonucleotide stability. For example, Turner and coworkers used oligonucleotides of defined sequence to analyze absorbance versus melting temperature to arrive at these thermodynamic parameters (ΔH , ΔS , and hence ΔG). To be more specific, inverse melting temperature was plotted against $\log(C_T/4)$ where C_T represents the total oligonucleotide strand concentration.

In cases where a good linear fit is observed, the thermodynamic parameters were obtained from the following equation.

$$\frac{1}{T_m} = \frac{2.3R}{\Delta H^0} \log\left(\frac{C_T}{4}\right) + \frac{\Delta S^0}{\Delta H^0}$$

In the above equation R is 1.987 cal/mol K. Once ΔH^0 and ΔS^0 were successfully obtained, ΔG^0 was calculated by making use of $\Delta G^0 = \Delta H^0 - T \Delta S^0$ at a temperature of 37 °C. In addition, the oligonucleotides were present in 1M NaCl. These values are recreated in Table 1 from reference 5. The error associated with these values of ΔG^0 determined by Turner et al. (4) is +/- 4%. Inclusive with the error the results are still applicable and provide structurally predictive output even when significantly rounded off for the purposes of integer multiples of core value (19).

Table 1. Thermodynamic parameters for propagation sequences
(Table recreated from Turner et al. (4))

Propagation sequence	ΔH^0 kcal/mol	ΔS^0 cal/mol K	$\Delta G^0_{37 \pm 4\%}$ kcal/mol
5'-AA-3' 3'-UU-5'	-6.6	-18.4	-0.9
AU UA	-5.7	-15.5	-0.9
UA AU	-8.1	-22.6	-1.1
CA GU	-10.5	-27.8	-1.8
CU GA	-7.6	-19.2	-1.7
GA CU	-13.3	-35.5	-2.3
GU CA	-10.2	-26.6	-2.1
CG GC	-8.0	-19.4	-2.0
GC CG	-14.2	-34.9	-3.4
GG CC	-12.2	-29.7	-2.9

Another important issue addressed by Turner and his colleagues (4) deals with the internal, bulge, and hairpin loops. These loops are depicted in Figure 5 along with an example for each.

Figure 5. Bulge, hairpin, and internal loops.
 Structure due to the loop[left], arbitrary example [right]

The change in Gibbs free energy associated with these loops was also determined by Turner and his coauthors (4) by using data from previous work and other publications. The free energy associated with these loops is recreated in Table 2 from Turner et al. (4).

**Table 2. Free energy change for loops in kcal/mol
(Recreated from Turner et al. (4))**

Loop Size	Internal loop	Bulge loop	Hairpin loop
1		3.3	
2	0.8	5.2	
3	1.3	6.0	7.4
4	1.7	6.7	5.9
5	2.1	7.4	4.4
6	2.5	8.2	4.3
7	2.6	9.1	4.1
8	2.8	10.0	4.1
9	3.1	10.5	4.2
10	3.6	11.0	4.3
12	4.4	11.8	4.9
14	5.1	12.5	5.6
16	5.6	13.0	6.1
18	6.2	13.6	6.7
20	6.6	14.0	7.1
25	7.6	15.0	8.1
30	8.4	18.8	8.9

Recall that Gibbs free energy change can be either positive or negative (or zero signifying the process is at equilibrium). In case of positive ΔG , the process is said to be endergonic meaning it is a non-spontaneous process, in the case of negative ΔG , the process is said to be exergonic signifying the process is spontaneous. These thermodynamic principles may be applied in aligning nucleotide sequences. Base pairing of two sequences involves their specific complementarity, where hydrogen bonds and base stacking energies play a major role. The concept of aligning sequences through the use of thermodynamic parameters computationally models this

complementarity using the same principles. If possible, this alignment strategy may have many advantages over some of the existing algorithms. Apart from the possible obviation of user defined parameters, Benedetti and coworkers (18) point out that the free energy change in a system affect the mutation rate of the nucleotide sequences, thus this approach may prove to be helpful in providing insight into evolution. Recall that transition mutation is defined as when one purine (A or G) is replaced by the other purine (G or A), and on the complementary strand pyrimidine (T or C) is replaced by the other pyrimidine (C or T). Transversion mutation on the other hand is defined as when one purine (A or G) is replaced by a pyrimidine (C or T), and on the complementary strand pyrimidine (T or C) is replaced by a purine (G or A). Transition mutations are observed to be more favorable over transversion mutations from an energetic standpoint. This is one possible explanation as to why the rate of transition mutation is higher than the rate of transversion mutation. This also suggests that thermodynamic parameters are likely to play a key role and these mutations may somehow be dependent on these energetics rather than randomness. The implication here is that thermodynamic parameters may be able to tell us more about mutation and evolution as well.

As far as sequence alignment via RNA structure energies is concerned, the underlying idea is that less and less energy will be required to denature the two strands as the number of mismatches increase. A study done by Lustig and coworkers (20) on a 93 base pair sequence of *mus musculus* m-RNA for thymus protein found that the free energy change increases significantly as the number of mismatches increase. The

basic procedure adopted in calculating the free energy change was to first convert the query sequence and subject sequences to RNA sequences by simply converting deoxythymidylic acid to uridylic acid. This is done to make the sequences compatible with the energy rules proposed by Turner (4). Next, the subject sequences were converted to their complementary form and flanked (one at a time) to the query sequence. Table 1 and Table 2 were used in computing the energy value for each pair and then summed. The conclusion of this study is promising for sequence alignment through the use of RNA structure energies.

In addition to making the nucleotide strands compatible with the energy rules, it is also important here to point out some of the advantages of converting the DNA to RNA strands. For example, the RNA secondary structures are well studied relative to those of DNA. In addition, RNA secondary structures are more predictive than those of DNA and as Figure 5 shows, RNA has the unique ability to fold on itself and through absence of base pairing can form loops which are much more studied than those of DNA.

3. RESEARCH OBJECTIVES

The objectives of this research were as following.

1. Catalogue 153 back-translated protein and r-RNA and t-RNA sequences.
2. Determine if any type of correlation exists between BLAST bits scores and the associated ΔG .

Cataloguing of the 153 DNA/RNA sequences involved a BLAST search, computation of ΔG values and storage of this information. An Access database for this purpose was created that housed the approximately 39,000 sequences resulting from the BLAST search of the 143 back-translated DNA sequences and the 10 RNA sequences. The primary objective of this research was to determine if there exists a correlation between the degree of alignment between two nucleotide sequences (as indicated by the BLAST bits scores) and the energy required to anneal the two strands. In other words, to study the relation between the Gibbs free energy change with respect to sequence alignment. The motivation for this project was the promising results of the preliminary study (20) that pointed to the likelihood of a correlation.

4. METHODS

A list of 143 proteins with known 3-dimensional structure were randomly chosen from Protein Data Bank in order to study the relation between energy required to anneal (or bifurcate) the two (back-translated) nucleotide strands (the query and the subject strands) and the degree of misalignment with respect to the query sequence. Also, 10 RNA (r-RNA and t-RNA) sequences were randomly chosen from PDB. These proteins/RNA are provided in Appendix A with their PDB identification numbers. An outline of the method is presented below followed by a further discussion on each step.

1. 143 Proteins from PDB with known 3-D structures were selected without any attempt to bias and based on plans of further possible application of the results of this research. The 10 RNA sequences were randomly chosen from PDB, 4 of which were t-RNA and 6 were r-RNA.
2. The proteins are then back-translated to the nucleotide sequence. The RNA sequences are used as given.
3. A BLAST search of GenBank was performed with each back-translated protein as query sequence as well as the 10 RNA sequences.
4. All resulting alignments are stored in a database (The GI #s and the BLAST scores associated with each protein).
5. All resulting DNA sequence alignments are then converted into RNA sequences in order to make them compatible with RNA structure energy rules.

6. All database sequences (subject sequences) were converted to their complementary form as well.
7. Java programming was used to determine the energy values associated with each alignment of query to subject sequence.
8. This data is stored in the database previously mentioned.
9. Correlation plots were used to determine if any relation among the energy changes and the BLAST scores exist.

The very first step in this procedure is to obtain the amino acid sequence of the protein in question. This is done through the use of Protein Data Bank (www.pdb.com), where the PDB ID number suffices to provide this information. In cases where multiple chains of the protein were encountered, Chain-A was always used unless otherwise stated. The amino acid sequence can be back-translated to the nucleotide sequence using the “standard” genetic code. However, it should be noted here that different organisms prefer different codons in encoding amino acids, therefore care must be taken in back-translating the amino acid chains. For this purpose whenever a protein was back-translated, organism specific codons were used in order to mitigate the problems of degeneracy of genetic code. Therefore, the 153 proteins used were back-translated applying organism specific codons. Also, organisms for which the preferred codons could not be found or were simply unknown (as there were six proteins as mentioned in the next chapter), the standard genetic code was applied. This part of the process is done by making use of the free online service provided by Entelechon (www.microbiology-direct.com) where organism specific codons may also be adopted

by specifying the organism's Latin name, or other programs such as DNASTrider may also be used here. As an example, this is illustrated with the protein Cytochrome C (horse heart). This protein will be used throughout this section to illustrate the adopted method.

PDB ID#: 1CRC

Amino acid sequence:

GDVEKGKKIFVQKCAQCHTVEKGGKHKTGPNLHGLFGRKTGQAPGFTYTDA
NKNKGITWKEETLMEYLENPKKYIPGTMIFAGIKKKTEREDLIAYLK KATNE

Nucleotide sequence:

GGG GAC GTA GAG AAG GGC AAG AAA ATT TTT GTC CAA AAG TGC GCG
CAG TGT CAT ACA GTG GAA AAG GGC GGT AAG CAC AAG ACC GGC CCT
AAC CTC CAC GGC CTG TTC GGA AGA AAA ACA GGG CAG GCT CCC GGG
TTT ACA TAC ACG GAC GCC AAT AAA AAC AAG GGT ATA ACT TGG AAA
GAG GAG ACC CTG ATG GAG TAC CTG GAA AAC CCA AAA AAG TAT ATC
CCA GGA ACC AAA ATG ATT TTC GCC GGC ATC AAG AAG AAA ACC GAG
AGG GAA GAT CTT ATC GCC TAC TTG AAG AAG GCA ACT AAT GAG

Next, BLAST search of GenBank (www.ncbi.nlm.nih.gov) is performed using the back-translated nucleotide sequences. Except for setting the number of alignments to maximum, the default settings in the Standard nucleotide-nucleotide BLAST search are used for all 153 query sequences. The default scoring matrix of BLASTN is BLOSUM62. This results in approximately two hundred hits on average per nucleotide query sequence. Therefore approximately 39,000 sequences were catalogued and Access was used to create a database to store this data. For Cytochrome C one of the hits from BLAST is shown for illustrative purposes. Note that redundancy is not present in these searches, meaning the query sequence is never returned as a subject sequence for a perfect match.

BLAST output (only showing one hit):

Score = 38.2 bits (19), Expect = 5.7
Identities = 22/23 (95%)
Strand = Plus / Minus

```
Query: 45      gcagtgtcatacagtggaaaagg 67
             |||||  |||||
Sbjct: 33167 gcagtgtcttacagtggaaaagg 33145
```

Note that BLAST attaches a score (in bits) that signifies the degree of alignment for each subject sequence; these scores may be used for comparison purposes among other BLAST scores. The alignments are provided in descending order according to these scores. Also observe that the output is a local alignment (as expected) since a segment of the query sequence is aligned to the subject sequence instead of aligning the query sequence as a whole.

Java programming was mainly used in applying the energy rules. This involved conversion of all DNA alignments obtained from the BLAST search to RNA strands (which is done simply by converting all the thymine residues to uracil residues) in order to make them compatible with the energy rules set forth in the literature review section (4). All subject sequences were also converted to their complementary form as well.

Perl programming was used to extract some of the pertinent information (such as GI #s, BLAST bits scores, and ΔG values) from the final output file obtained through Java. Perl was also used in determining the GC content and related information.

Another alignment of Cytocrome C is used again to illustrate the energy calculations.

Query: u g c g c a c a g u g u c a

Energ: -1.8 -3.4 -2.0 -3.4 0.8 -1.8 0.8 -1.8 -2.1 -2.3 -1.8

Sbjct: a c g c g a g u u a c a g u

Query: u a c c g u a g a a a a a g

Energ: -0.9 -1.1 -2.1 -2.9 -2.0 -2.1 0.8 -2.3 -0.9 -0.9 -0.9 -0.9 -1.7

Sbjct: a u g g c a a c u u u u u c

Query: g c g g

Energ: -2.9 -3.4 -2.0 -2.9

Sbjct: c g c c

The total energy score is -47.9 kcal/mol

The number of matches: 29 / 32, Total gaps 0

Score = 40.1 bits (20)

Identities = 29/32 (90%)

Note that the subject sequence has been transformed into its complementary form and the duplex energy values (with penalties for loops) are summed to provide the ΔG of the considered strand. Also, during this process all alignments given by BLAST were considered, therefore whenever several alignments per GI # were encountered they were treated as any other alignment.

The ΔG_{37}^0 calculated for this alignment is -47.9 kcal/mol ($\pm 1.9 \text{ kcal/mol}$) as shown above. In this manner all alignments resulting from the BLAST search are used to calculate the energy associated with each query - subject sequence alignment. In

order to determine if there exists a correlation between the BLAST bits score and the Gibbs free energy change associated with the corresponding alignments, correlation plots are used with the energy change (in kcal/mol) plotted on the y-axis and the BLAST bits scores plotted on the x-axis.

The equipment required for this project was essentially a computer (with ~ 4 Gigabytes of free hard drive space) and access to the World Wide Web to access the databases. In addition, a database of the acquired data in the duration of the project was created using Microsoft Access. Java was used for all energy calculations and Perl was used to extract pertinent information from these files.

5. RESULTS AND DISCUSSION

Due to the fact that all resulting alignments from the BLAST searches were taken into account in energy determination, the total number of alignments obtained reached 38,995. This is also due to setting the alignments/descriptions parameter of BLAST to maximum. Therefore there are 38,995 data points available for the correlation plots. Before proceeding on with the obtained results, there are a few items worth pointing out.

1. No deviations were made/introduced in following the method as delineated in the method section.
2. No alignments were rejected, all alignments resulting from BLAST search were taken into account. This includes the several alignments observed per GI# as well.
3. All BLAST scores and corresponding energy values for each protein were placed on a correlation plot, therefore 153 correlation plots were created.
4. The GC and AU content was calculated from all 143 back-translated proteins and the 10 RNA sequences and is presented in this section.

It is also important to point out here that during the amino acid sequence acquisition from PDB, multiple chains of the protein were often encountered. In these cases the very first chain (often Chain-A) was almost always taken for this study; if otherwise, the chain was identified and stored alongside data pertaining to that

identifier. The main reason for not choosing Chain-A in these observed rare cases was the presence of multiple unidentified amino acids (Xs) in the sequence.

Also, due to the difficulty encountered in obtaining organism specific codons for 1agm, 1agx, 1alc, 1amn, 3lzm, and 7pcy, the standard genetic code was applied to back-translate these six specific proteins. The total number of alignments resulting from these six proteins were 199, hence they do not have any significant effect on the outcome.

The total number of residues comprising the 153 query sequences were 115,559. In all 115,559 residues only A, C, G, and Ts (and Us in place of Ts in RNA strands) were observed, some of the other uncommon residues were not observed. However, Ns (symbolizing unspecified residues) and Rs (symbolizing unspecified purine residues) were rarely encountered in the BLAST searches. These were consistently treated as gaps by replacing them with the gap character (-) and the Java code was written to recognize the gap character and treat it as such.

The BLAST hits per query sequence ranged from just 8 to 8,128 alignments. On average 100-250 hits were observed for most proteins. The least number of hits (of 8) were observed for the protein 1a59 (Citrate Synthase from Antarctic bacterium DS2-3R, 1134bp) and the most (of 8,128) for r-RNA sequence 1c2w (23S Ribosomal RNA- Chain B, 2,904bp). There were approximately 5 other proteins that resulted in about 10 alignments and approximately 10 proteins/RNA that resulted in over 1,000 alignments. The remaining of the 153 proteins/RNA were mostly between the range of 20 – 600 alignments.

The 153 correlation plots contained in Appendix B are essentially the results of this research. Due to the enormity in size, the 38,995 alignments and computed energy scores are not listed. The correlation plots however represent the 38,995 data points for each protein/RNA. For each correlation plot the BLAST score is plotted on the x-axis and the computed ΔG values on the y-axis. In addition, a linear curve is fit to the data upon observing that most plots appear to be linear. An R^2 value as well as the linear equation for the curve fit is also obtained and shown for each correlation plot. Figure 6 below shows a typical correlation plot; observe that the format of the plot is kept consistent for all 153 figures. Note that the length of the back-translated sequence as well as of RNA molecule is given in parenthesis in plot title (omitted here).

Figure 6. Correlation plot for back-translated Modified α_1 -Antitrypsin (7api)

Here, it is worth pointing out that although the above plot appears to have points on the order of ~20, but the actual total number of points (representing number of alignments) are 98. Many points are overlapping points and this is true for almost

all plots given in Appendix-II, therefore the total number of alignments are also given in Appendix-II for each plot.

Figure 7 shows some representative plots of the 153. The first plot is of the worst R^2 value; the second correlation plot is of the average R^2 value which is observed to be typical of all 153 plots (computed to be 0.83). Lastly the third plot is of one of the highest R^2 values obtained.

Figure 7. Plots of lowest, average, and highest R^2 value (from top to bottom)

Note that the overall trend in all correlation plots is not only linear but also of negative slope (see Appendix B also). The energy values decrease with increasing BLAST scores. This is not surprising since higher degree of alignment (with fewer mismatches to form loops) is expected to result in lower total energy value as set forth by Turner et al. (4). However, it is interesting to observe that there appears to be a linear correlation between the energy values and BLAST scores.

Furthermore, Figure 8 shows the aggregate correlation plot of all 38,995 alignments. The format of the aggregate plot in this figure as well as other aggregate plots to come is consistent with the correlation plots. Observe that the data itself indicates linearity, which is observed on the discrete plots as well. Also note that the R^2 is 0.92, which indicates a good linear fit to the data and that the slope is -1.07 indicating a strong correlation between the BLAST bits score and the corresponding ΔG values.

Figure 8. Aggregate correlation plot of all 38,995 alignments

Due to the large number of alignments (8,128) resulting from 1c2w r-RNA sequence, it alone has an enormous effect on the aggregate correlation plot since 1c2w by itself represents 20.8% of these total 38,995 alignments. Therefore, a correlation plot exclusive of 1c2w is constructed and is shown in Figure 9.

Figure 9. Aggregate correlation plot excluding 1c2w

Observe that the slope has changed from -1.07 to -2.25 . Also note that R^2 has dropped from 0.92 to 0.74 . A discussion on whether this R^2 indicates a strong correlation or not and on the significance in change in slope is reserved for later.

Next, Figure 10 shows the aggregate correlation plot of all alignments exclusive 1c2w in two parts, a low scoring region and a high scoring region correlation plot with respect to the BLAST scores. Figure 10 is a magnified version of Figure 9, where more detail may be observed.

Figure 10. Aggregate correlation plots of low and high scoring regions (exclusive lc2w)

So far, aggregate correlation plots have been shown with results from both proteins and RNA sequences mixed and no differentiation has been made based upon whether the results are from RNA or back-translated amino acid query sequences. Figure 11 shows two correlation plots, the first being that for the protein derived sequences only and the second being that for the RNA sequences only. The RNA alignments plot in Figure 11 only shows the region below BLAST score of 2,000 as there are only 94 alignments observed above this region that span out to a BLAST score of 6,000.

The very first observation that may be made from Figure 11 is the proximity of r-RNA and t-RNA alignment data to the linear fit relative to that of protein alignment data. This observation may be explained due to the simple fact that the energy calculations are specifically for RNA structures. Also note that in converting the DNA molecules to RNA (by changing all T residues to U), RNA is in effect used to model DNA hybridization.

Conversion of the DNA chains to RNA results in RNA forming double strands as those observed for DNA. This allows for the unique properties of RNA to be used in modeling DNA hybridization. The formation of loops, for example, are due to the presence of gaps and mismatches and therefore the RNA structure energies associated with these loops may be applied. Here the complementary matching regions are awarded negative ΔG (as per RNA duplex ΔG values (4)) and the mismatching regions are modeled as RNA secondary structure loops and hence are given a positive ΔG . As

further discussion of these results will reveal, the use of RNA secondary structure energies appears to provide a very unique and a beneficial approach in modeling gaps.

Figure 11. Aggregate correlation plots for results from amino acid and RNA sequences, separately

Observe from Figure 11 that the slope of the proteins only plot (-2.4) is much steeper than that of the RNA alignments (-1.3). For RNA sequences there appears to be almost a one-to-one relation between the BLAST bits score and the corresponding ΔG . But for protein derived sequences, however, since the slope is much steeper, a method comprised of these energy rules for sequence alignment would result in higher sensitivity relative to the BLAST scoring scheme. Sensitivity here describes that a small change in BLAST score results in a large change of the associated ΔG value, so the energy rules approach is likely to be capable of finding more alignments or small alignments than BLAST. Hence we may say that a method using these energy rules provides a more sensitive approach for alignment than BLAST.

Upon closer inspection of the RNA correlation plot in Figure 11, observe that the section below 5,00 BLAST score region has the most data under the linear fit and since the slope of the fit is already -1.3, we may say that especially in this region the energy rules appear to provide more sensitivity for alignment than BLAST (the slope in this region is -1.5). On the same note, close to the 6,000 BLAST score region, observe from Figure 8 that the curve appears to be “flattening,” therefore here the energy rules seem to lose their sensitivity. This, however, should not be given too much attention since this observation is due to a single query sequence (1c2w). Observe from Figure 9 (aggregate plot exclusive of 1c2w, slope of -2.25) that although many points are above the linear fit but majority of the data seems to be lying under the fitted curve. In the proximity of the 1,000 BLAST score region in Figure 9 one could argue that the slope in that local region is much lower and hence the same

argument may be applied that the energy rules are not more sensitive in that region. Also, the high BLAST score correlation plot of Figure 10 shows the aggregate plot (without 1c2w) with BLAST scores above 400, note that the slope is -0.49 . Therefore the energy rules are definitely not as sensitive here, but also note that this plot is comprised of only 82 data points (82 alignments) and the R^2 is 0.13 so any bold statements should be avoided here. Nevertheless, it should be obvious (especially in case of protein-derived sequences), from some of the localized data lying below the linear fit and the overall slope being much less than -1 seem to indicate a higher sensitivity for alignment is provided by energy rules than by BLAST, this should be clear from both Figure 9 and 11.

Observe from the discrete and aggregate correlation plots that the ΔG decreases with increasing BLAST scores. Note that in all plots we observe a negative slope. This is expected as the ΔG will decrease with increasing alignment length. Furthermore, as seen in Figure 11 with back-translated protein alignments the slope is much less than -1 . The significance of this fact may be exploited in sequence alignment as following. Since a steeper slope here suggests that RNA secondary structure energies approach is relatively more sensitive than BLAST scoring scheme, therefore as these energy rules are incorporated into BLAST or other existing search/alignment methods (such as Smith-Waterman) the sensitivity in sequence alignment is likely to improve. In addition, Smith-Waterman is an ideal algorithm that can be modified to take advantage of this approach since both methods assign a numerical value to duplex pairs.

One approach to this incorporation is as following, the Smith-Waterman scoring scheme may be preserved but the numerical values of the duplex pairs from these energy rules may be substituted for those currently used in Smith-Waterman. The matches would result in negative ΔG values whereas the mismatches (the loops) are assigned positive ΔG values (penalties). In addition to transforming both the query and subject DNA sequences to RNA sequences, also note that the subject sequences would need to be converted to their complementary form. To elaborate further on the incorporation of energy rules in Smith-Waterman, consider the possibility of adding the ΔG values of each duplex pair to the existing numerical values used by Smith-Waterman scoring scheme. This simple operation allows for the existing Smith-Waterman to be altered to take advantage of what the RNA structure energies approach has to offer and retain the unique qualities of standard Smith-Waterman as well. This addition should introduce the advantages observed in the use of these energy rules, namely the higher sensitivity. Various other combinations and possibilities may be explored as this is just the nascent stage of this type of exploration with the RNA secondary structure energies. Once these energy rules are incorporated into Smith-Waterman, a test may be performed to evaluate the performance of standard Smith-Waterman versus the RNA-based Smith-Waterman.

Next, in order to test the statistical validity and strength of the correlation, a statistical check was made. Table 3 presents the statistical probability (P) of exceeding the linear correlation coefficient (R) for each of the 153 query sequences, and it also presents the P-value for some of the figures presented in this section (the correlation

plot of all 38,995 alignments, for example). In the latter, the aggregate correlation plots are identified by the figure number. Table 3 also summarizes the number of alignments observed for each of the query sequences as these are considered in determining P. The average boundary P-value of 153 proteins/RNA is determined to be 0.001, the worst is observed for protein lagx with R of 0.73 and 12 alignments resulting in P-value of 0.007. A boundary P-value of 0.001 is observed for many proteins/RNA discrete correlation plots as well as for all aggregate correlation plots, excluding the 82-alignment aggregate plot of Figure-10, shown up to this point.

The P-values may be interpreted as following: the average boundary statistical probability of 0.001 for all 153 proteins/RNA means that there is approximately 1 in a 1,000 chance that a random set of data will generate the same linear correlation coefficient of 0.91 (recall that the average R^2 is observed to be 0.83, therefore $\sqrt{R^2}$ is 0.91). Also note that generally a P-value of 0.005 is taken to be sufficient to conclude that statistically the correlation is valid and significant. The only P-value above 0.001 of any discrete protein derived or RNA correlation plot is for protein lagx (P of 0.007). Observe that the P-values for this discrete protein is high only due to the small number of alignments observed (which is 8) as the linear correlation coefficient is 0.73. Other correlation plots have this R-value but due to the larger set of alignments the P-values are much lower. Also, this is one protein from a list of 153, and as the average boundary P-value of 0.001 for all 153 correlations signifies the statistical significance of this correlation, one should not devote much effort to this single protein. Therefore the current boundary P-value of 0.001 is more than acceptable to

state that there is a strongly significant correlation between the BLAST bits score and the corresponding ΔG . In addition, consider the aggregate plot of all 38,995 alignments. The boundary P-value is determined to be 0.001 (the actual value is much lower), and this is true for most aggregate plots shown so far (including Figure 9, the aggregate plot without 1c2w). This provides further statistical evidence to support the above claim.

Table 3. P-values for the 153 query sequences and for some aggregate plots.
 (# below represents the number of alignments, * means the P-value is < 0.001, table continued on next page)

ID	Protein Source	R	#	P	ID	Protein Source	R	#	P
1a3c	Bacillus subtilis	0.99	15	0.001*	1ubq	Homo sapien	0.88	1584	0.001*
1a3s	Homo sapien	0.94	111	0.001*	2act	Actinidia chinensis	0.87	81	0.001*
1a4S	Saccharomyces cerevisiae	0.88	28	0.001*	2cab	Homo sapien	0.84	41	0.001*
1a59	Antarctic Bacterium	0.99	8	0.001*	2cdv	Desulfovibrio vulgaris	0.97	25	0.001*
1a5z	Thermotoga Maritima	0.99	19	0.001*	2cga	Bos taurus	0.93	48	0.001*
1a6q	Homo sapien	0.86	126	0.001*	2ci2	Hordeum vulgare	0.96	13	0.001*
1aat	Gallus gallus	0.92	24	0.001*	2cpp	Pseudomonas putida	0.97	17	0.001*
1ab4	E. coli	0.95	282	0.001*	2cro	Bacteriophage 434	0.99	9	0.001*
1abm	Homo sapien	0.87	137	0.001*	2cts	Sus scrofa	0.92	269	0.001*
1acb	Bos taurus	0.89	49	0.001*	2cyp	Saccharomyces cerevisiae	0.98	23	0.001*
1adb	Equus caballus	0.95	195	0.001*	2fb4	Homo sapien	0.97	772	0.001*
1add	Mus musculus	0.97	80	0.001*	2fbj	Mus musculus	0.86	228	0.001*
1adi	E. coli	0.99	49	0.001*	2fxb	Bacillus thermoproteolyticus	0.86	22	0.001*
1ae4	Sus Scrofa	0.95	127	0.001*	2gbp	E. coli	0.99	23	0.001*
1af3	Rattus	0.84	79	0.001*	2gcr	Bos taurus	0.75	106	0.001*
1agm	Aspergillus awamori	0.85	42	0.001*	2gn5	filamentous bacteriophage	0.99	54	0.001*
1agx	Acinetobacter Glutaminasificans	0.73	12	0.007	2hfl	Mus musculus	0.66	849	0.001*
1ahn	E. coli	0.97	63	0.001*	2hla	Homo sapien	0.82	1478	0.001*
1ai2	E. coli	0.98	156	0.001*	2hmg	Influenza virus	0.71	807	0.001*
1ak2	Bos Taurus	0.89	48	0.001*	2lbp	E. coli	0.98	42	0.001*
1ako	E. coli	0.99	26	0.001*	2ldx	Mus musculus	0.92	132	0.001*
1al8	Spinacia oleracea	0.88	71	0.001*	2liv	E. coli	0.86	66	0.001*
1alc	Papio hamadryas cynocephalus	0.92	27	0.001*	2ltm	Pisum sativum	0.83	30	0.001*
1aln	E. coli	0.99	14	0.001*	2lz2	Meleagris gallopav	0.71	20	0.001*
1als	Homo sapien	0.99	51	0.001*	2lzt	Gallus gallus	0.69	33	0.001*
1amn	Toppedo Californica	0.98	58	0.001*	2mev	Mengo virus	0.65	79	0.001*
1amp	Aeromonas proteolytica	0.99	20	0.001*	2mhb	Equus caballus	0.94	351	0.001*
1an9	Sus Scrofa	0.94	71	0.001*	2pab	Homo sapien	0.95	55	0.001*
1ao5	Mus musculus	0.92	177	0.001*	2pka	Sus scrofa	0.87	37	0.001*
1aob	E. coli	0.97	45	0.001*	2plv	Homo sapien	0.93	335	0.001*
1aqh	Alteromonas Haloplanctis	0.99	48	0.001*	2prk	Triticachium album	0.99	43	0.001*
1azi	Equus caballus	0.94	35	0.001*	2ptn	Bos taurus	0.80	250	0.001*
1crc	Equus caballus	0.86	189	0.001*	2rhe	Homo sapien	0.57	196	0.001*
1lzl	Homo sapien	0.95	94	0.001*	2rn2	E. coli	0.99	30	0.001*
1omd	Rattus	0.96	28	0.001*	2rsp	Rous sarcoma virus	0.97	40	0.001*
1rbp	Homo sapien	0.98	49	0.001*	2sga	Streptomyces griseus	0.99	33	0.001*
1rhd	Bos taurus	0.85	40	0.001*	2sni	Bacillus amyloliquefaciens	0.91	113	0.001*
1sgt	Streptomyces griseus	0.98	138	0.001*	2ssi	Streptomyces albogriseolus	0.99	64	0.001*
1tgn	Bos taurus	0.73	199	0.001*	2stv	Tobacco necrosis virus	0.93	21	0.001*
1ton	Rattus	0.89	91	0.001*	2taa	Aspergillus oryzae	0.92	86	0.001*

Table 3 continued from previous page

ID	Protein Source	R	#	P	ID	Protein Source	R	#	P
2tbv	Tomato bushy stunt virus	0.89	22	0.001 *	4tnc	Gallus gallus	0.80	69	0.001 *
2tim	Trypanosoma brucei brucei	0.96	90	0.001 *	4xia	Arthrobacter sp.	0.99	34	0.001 *
2tmv	Tobacco mosaic virus	0.85	105	0.001 *	5acn	Sus scrofa	0.95	178	0.001 *
2trx	E. coli	0.99	45	0.001 *	5cha	Bos taurus	0.89	42	0.001 *
2tsc	E. coli	0.96	76	0.001 *	5cpa	Bos taurus	0.95	68	0.001 *
2tsl	Bacillus stearothermophilus	0.97	86	0.001 *	5cpv	Cyprinus carpio	0.99	22	0.001 *
2wrp	E. coli	0.92	24	0.001 *	5cts	Gallus gallus	0.77	84	0.001 *
3adk	Sus scrofa	0.92	47	0.001 *	5ldh	Sus scrofa	0.77	279	0.001 *
3app	Penicillium janthinellum	0.79	22	0.001 *	5rsa	Bos taurus	0.95	69	0.001 *
3blm	Staphylococcus aureus	0.99	109	0.001 *	5rub	Rhodospirillum rubrum	0.98	144	0.001 *
3cla	E. coli	0.95	255	0.001 *	5tnc	Meleagris gallopavo	0.97	70	0.001 *
3cln	Rattus	0.92	1221	0.001 *	6ldh	Squalus acanthias	0.63	165	0.001 *
3cna	Canavalia ensiformis	0.93	24	0.001 *	6xia	Streptomyces albus	0.99	77	0.001 *
3dfr	Lactobacillus casei	0.99	17	0.001 *	7api	Homo sapien	0.99	98	0.001 *
3est	Sus scrofa	0.90	44	0.001 *	7cat	Bos taurus	0.85	186	0.001 *
3fab	Homo sapien	0.88	1016	0.001 *	7pcy	Enteromorpha prolifera	0.81	24	0.001 *
3gap	E. coli	0.96	42	0.001 *	8adh	Equus caballus	0.91	174	0.001 *
3gbp	Salmonella typhimurium	0.97	301	0.001 *	8atc	E. coli	0.97	37	0.001 *
3gpd	Homo sapien	0.63	984	0.001 *	8dfr	Gallus gallus	0.78	40	0.001 *
3grs	Homo sapien	0.89	61	0.001 *	9pap	Carica papaya	0.88	36	0.001 *
3hfm	Mus musculus	0.79	1106	0.001 *	9wga	Triticum aestivum	0.98	30	0.001 *
3icb	Bos taurus	0.73	28	0.001 *	1c2w	E. coli	0.99	8128	0.001 *
3lzm	coliphage T4	0.92	36	0.001 *	1c2x	E. coli	0.98	583	0.001 *
3mcg	Homo sapien	0.89	667	0.001 *	1co4	Candida glabrata	0.87	1057	0.001 *
3plk	Bacillus stearothermophilus	0.98	21	0.001 *	1emu	Thermus Thermophilus	0.97	2477	0.001 *
3pgk	Saccharomyces cerevisiae	0.96	67	0.001 *	1euy	E. coli	0.94	138	0.001 *
3pgn	Saccharomyces cerevisiae	0.94	32	0.001 *	1exd	E. coli	0.97	101	0.001 *
3psg	Sus scrofa	0.88	124	0.001 *	1ffy	Staphylococcus aureus	0.91	1021	0.001 *
3m3	Bos taurus	0.82	96	0.001 *	1fgo	Glycine max	0.98	1136	0.001 *
3mt	Aspergillus oryzae	0.95	15	0.001 *	1qu3	E. coli	0.91	1021	0.001 *
3rp2	Rattus	0.97	19	0.006	357d	E. coli	0.96	218	0.001 *
3sgb	Streptomyces griseus	0.97	30	0.001 *					
3tln	Bacillus thermoproteolyticus	0.97	46	0.001 *					
4ape	Cryptosporidia parvifilum	0.89	23	0.001 *		Fig 8	0.92	38995	0.001 *
4dfr	E. coli	0.95	41	0.001 *		Fig 9	0.74	30867	0.001 *
4fab	Mus musculus	0.83	1120	0.001 *		Fig10(low)	0.61	30785	0.001 *
4fkn	Clostridium beijerinckii	0.95	139	0.001 *		Fig10(high)	0.13	82	0.244
4hvp	Human immunodeficiency virus type 1	0.89	1076	0.001 *		Fig 11Prot.	0.77	23115	0.001 *
4mdh	Sus scrofa	0.94	98	0.001 *		Fig 11RNA	0.95	15786	0.001 *
4pep	Sus scrofa	0.79	94	0.001 *					
4rhv	Human rhinovirus 14	0.95	84	0.001 *					
4sbv	Southern bean mosaic virus	0.98	10	0.001 *					

Furthermore, in order to study the behavior of BLAST scoring scheme versus the use of energy rules in sequence alignment for different organisms, the 143 proteins were filtered for those with source *Escherichia coli* from those of *Homo sapien* source. 18 proteins were found to be of *Homo sapien* source and 18 of *E. coli* source. The list of PDB ID numbers under each source are identified in Appendix C.

Figure 12 shows two correlation plots, the first for proteins with source *E. coli* and the second for proteins with *Homo sapien* source. Observe that both appear to be identical, except for a very small change in slope. The slope for *Homo sapien* source plot is -2.60 and for *E. coli* source plot is -2.63 . The R^2 values are also different but that is only a measure of linear correlation and it is not the objective here to debate whether a better fit for *E. coli* is due to the energy rules or BLAST. It's more important to determine whether either of these methods treat different types of organisms in a different manner or not.

Figure 12. Correlation plots classified by two different sources

It is impossible to answer this newly broached question by simply observing the two correlation plots in Figure 12. Here it is important to make the observation that one must compare the two methods separately. The conventional correlation plots that have been used so far relate the BLAST scores to the corresponding ΔG , therefore both are made interdependent. However, both BLAST scores as well as ΔG may be compared separately via sequence alignment lengths. This provides an independent intermediary variable, which is common in both methods and yet does not influence the outcome in either method's favor. Figure 13 shows these two plots, the first is of BLAST scores versus the alignment length and the second of ΔG versus alignment length.

It should be kept in mind that this data is extracted from the same 143 proteins and the alignment lengths are common between both plots. Relatively, observe from Figure 13 that Blast scores differ considerably from *Homo sapien* source proteins and proteins from *E. coli*. The BLAST scores appear to diverge with alignment length. The energy rules on the other hand provide a scale that is completely oblivious as to the type of organism the proteins are from. This shows that the BLAST scoring scheme seems to treat different type of organisms differently but the energy rules do not. The BLAST scores for prokaryotes and mammals diverge with alignment length but the energy aligned ΔG values are constant. This indicates that the energy rules are relatively more universal and generalized than BLAST. In addition, this feature of energy rules should be observed in addition to the higher sensitivity when incorporated into Smith-Waterman, for example. Also note from Figure 13 that the

error associated with the linear fit for the BLAST score versus alignment length plot does not allow for any overlap of the two linear fits. The overlap is only observed near the limited region where the actual linear fits overlap themselves.

Figure 13. Comparison of BLAST scoring versus the energy rules

These findings seem to indicate that it is most likely that the better results are attained because of superior evaluation of gap penalties using the physically based model results of RNA secondary structure energies. Therefore, in addition to being more sensitive and generalized, the energy rules provide a more realistic approach to modeling gaps. This is an important point to note since the treatment of gaps has been the most vexing problem for researchers in sequence alignment.

One of the concerns with the use of RNA structure energies is the relatively low value of ΔG assigned to the GC and GG duplex pairs and the high ΔG values for the duplex pairs involving A and U. Duplexes composed purely of AU content (the first three in Table 1) have a ΔG on the order of -1 kcal/mol. Also observe that the duplexes 5'-GC-3' and 5'-GG-3' (flanking the complementary residues), have a ΔG on the order of -3 kcal/mol. All other propagation sequences (duplex pairs as shown in Table 1) reserve a ΔG of about -2 kcal/mol. The physical fact that a lower ΔG is observed in the last two duplexes is termed "GC pair bias" since an alignment with high G+C content is likely to result in a lower ΔG than the same alignment length with low G+C content. On the same note, a similar argument may be applied for the strands with high A+U content since now the resulting ΔG will be higher than those with low A+U content. For clarification purposes the term bias has been used here since these representative ΔG values are considered as energy "scores" in sequence alignment, therefore an immediate question arises – why should the energy "score" be higher (or lower) for two different alignments solely based on the residue content when the sequence lengths and percent mismatches are the same? At the same time,

the fact that the energy values are different in this example is a statement on the thermodynamic stability of the alignments, the lower the ΔG the higher the alignment stability, hence there is a (potentially useful) physical significance to these “scores.”

To address this issue the G+C content of the 143 back-translated amino acid sequences was calculated with the aid of Perl programming (all codes are presented in Appendix D). The percent G+C content was found to be 51.0 and the A+U content to be the remaining 49.0. The total number of residues were 111,399. For the 10 RNA sequences, the GC content was determined to be 54.8% and the AU content was the remaining 45.2%. The total number of residues were 4,160. Therefore the overall composition is half and half of G+C and A+U residues. Furthermore, the AA+UU, 5'-AU-3', 5'-UA-3', GG+CC, and 5'-GC-3' composition compatible with the energy rules as set forth by Turner et al. (4) was determined. Observe that the first three pairs listed above have a ΔG of approximately -1 kcal/mol and the last two of about -3 kcal/mol. For purposes of clarity, what is meant by the propagation sequences being “compatible” with the energy rules is as following. In case of GG pairs, for example, wherever a sequence with three consecutive G residues is observed it is counted as 2 pairs of GG, and a sequence with four consecutive G residues as 3 pairs and so on. The composition of these propagation sequences is given in Figure 14.

Figure 14. Distribution of duplexes with ΔG of -1 and -3 kcal/mol
(In the key ΔG values are shown in parenthesis in kcal/mol)

Observe that 46% of the propagation sequences (compatible with the energy rules) have a ΔG of -3 kcal/mol and -1 kcal/mol for the remaining 54%. Note that the -3 kcal/mol is more than offset by the -1 kcal/mol propagation sequences. If we compute the overall ΔG according to Figure 14, we obtain:

$$(-0.9)(0.30)+(-0.9)(0.14)+(-1.1)(0.11)+(-2.9)(0.30)+(-3.4)(0.15) = -1.9 \text{ kcal/mol}$$

Observe that this ΔG is approximately the same as that of all other propagation sequences. This signifies some sort of complementary balance.

The total number of residues for the 153 query sequences are 115,559 and these 153 sequences can be treated as a completely random selection pertaining to the GC bias. As this is a fairly large set of data, these results are likely to be typical of the general genome content. However, in the special cases where the G+C content in specific genes is high this GC bias will remain in the sense that the ΔG will be lowered. Similarly where the A+U (or A+T) content is observed to be high, the ΔG will be relatively high. As these are natural parameters it is impossible to circumvent

this bias using the nearest neighbor model as is done here. In general, this is not as large of an issue as it appears to be. As mentioned earlier, in this specific set of data the G+C content is approximately 50% as is the A+U content. A critical point to this discussion is the uniformity of these propagation sequences in the 153 query sequences. Upon inspection of the query sequences these residues are well distributed, therefore the -3 kcal/mol offset the -1 kcal/mol propagation sequences uniformly and vice versa and each query sequence is well represented by Figure 14.

This research also included creation of a database using Access. The database contains the GI #s, the Blast scores and the energy values associated with the corresponding alignments for each sequence. The purpose of the database is to conveniently recall all alignments (GI #s) by producing a query using the PDB identifiers for any future research. Individual GI #s may also be recalled by initiating a query by the GI #. In addition, all original BLAST files (153 total) were stored on a removable hard disk for future reference.

6. CONCLUSIONS

Upon studying the results attained from the 153 query sequences used for BLAST searches where RNA secondary structure energies as set forth by Turner et al. were used to compute the corresponding ΔG s, there is a statistically significant linear correlation between BLAST bits score and the ΔG required to anneal/bifurcate the two aligned sequences. The average linear correlation coefficient (R) was determined to be 0.91 and the average (boundary) probability of exceeding this R was calculated to be 0.001. In addition, the R-value of the aggregate plot of all 38,995 alignments was 0.92 and the boundary P-value was 0.001. The slope of less than -1 of the aggregate correlation plots for protein-derived sequences and RNA seem to show that the energy rules approach appears to be more sensitive than the BLAST scoring scheme.

Furthermore, the RNA secondary structure energy rules are determined to be less dependent on the type of organism than BLAST and are therefore more universal. Lastly, the GC pair bias is observed to be offset by the AU pairs.

REFERENCES

1. Waterman M. S. and Eggert M., *A new algorithm for best sequence alignments with application to tRNA-rRNA comparisons*. J. Mol. Biol., 1987, 197, pg-723.
2. Pearson W. R. and Lipman D.J., *Improved tools for biological sequence comparison*. Proc. Natl. Acad. Sci. U.S.A., 1988, 85, pg-2444.
3. Altschul, Gish, Miller, Myers and Lipman, *Basic Local Alignment Search Tool*, J. Mol. Biol., 1990, 215, pg-413.
4. Turner, Sugimoto and Freier, *RNA structure prediction*, Ann. Rev. Biophys. Biophys. Chem., 1988, 17, pg-167.
5. Nicholas H., Dearfield D. and Ropelewski A., *Strategies for searching sequence databases*. Biotechniques, 28, 2000, pg-1174.
6. Baxevanis D. A. and Ouellette B., *Bioinformatics: A practical guide to the analysis of genes and proteins*. 1998.
7. Needleman S. and Wunsch C., *A general method applicable to the search for similarities in the amino acid sequences of two proteins*. J. Mol. Genetics, 48, 1970, pg-443.
8. Sellers P., *Pattern recognition in genetic sequences*, Proc. Natl. Acad. Sci. U.S.A., 76, 1979, pg-3041.
9. Shapaer E., Robinson M., Yee D., Candlin J., Mines R. and Hunkapiller T., *Sensitivity and selectivity in protein similarity sequences: A comparison of Smith-Waterman in hardware to BLAST and FastA*. Genomics, 38, 1996, pg-179.
10. Pearson W., *Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FastA algorithms*. Genomics, 11, 1991, pg-635.
11. Torbjorn Rognes, *ParAlign: a parallel sequence alignment algorithm for rapid and sensitive database searches*, Nucl. Acids Res., 29, 2001, pg-1647.
12. Straden R., *An interactive graphics program for comparing and aligning nucleic acid and amino acid sequences*. Nucl. Acids Res., 10, 1982, pg-2951.

13. Smith R. F. and Smith T. F., *Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modeling*, Prot. Eng., 5, 1992, pg-35.
14. Gupta S. K., Kececioğlu J. D. and Schaffer A., *Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment*, J. Comp. Biol., 23, 1995, pg-459.
15. Feng D. F. and Doolittle R. F., *Progressive sequence alignment as a prerequisite to correct phylogenetic trees*. J. Mol. Evol., 25, 1987, pg-351.
16. Morgenstern B., Dress A. and Warner T., *Multiple DNA and protein sequence alignment based on segment to segment comparison*, Proc. Natl. Acad. Sci. U.S.A., 93, 1996, pg-12098.
17. Voet, Voet and Pratt, *Fundamentals of Biochemistry*, Upgrade edition, Wiley, 2002, Ch-23.
18. Benedetti G., De Santis and Morosetti S., *A new method to find a set of energetically optimal RNA secondary structures*. Nucl. Acids Res., 17, 1989, pg-5149.
19. Lustig B., Covell G. and Jernign R., *Conformations of t-RNA: Base Pairing and Stacking*, J. Biomol. Struct. Dyn., 12, 1994, pg-935.
20. Lustig B., Nguyen A., Mitra S. and Rishiphatak D., *Sequence alignment using thermodynamic parameters*, Thirteenth annual California State University Biotechnology Symposium, 2001.

APPENDIX A

List of 153 PDB Identifiers (The italicized ID #s are of RNA sequences)

Protein Data Bank ID #s							
1a3c	1ako	1ubq	2ldx	2tbv	3hfm	4rhv	8dfr
1a3s	1al8	2act	2liv	2tim	3icb	4sbv	9pap
1a48	1alc	2cab	2ltm	2tmv	3lzm	4tnc	9wga
1a59	1aln	2cdv	2lz2	2trx	3mcg	4xia	<i>1c2w</i>
1a5z	1als	2cga	2lzt	2tsc	3pfk	5acn	<i>1c2x</i>
1a6q	1amn	2ci2	2mev	2tsl	3pgk	5cha	<i>1co4</i>
1aat	1amp	2cpp	2mhb	2wrp	3pgm	5cpa	<i>1emi</i>
1ab4	1an9	2cro	2pab	3adk	3psg	5cpv	<i>1euv</i>
1abm	1ao5	2cts	2pka	3app	3rn3	5cts	<i>1exd</i>
1acb	1aob	2cyp	2plv	3blm	3rnt	5ldh	<i>1ffy</i>
1adb	1aqh	2fb4	2prk	3cla	3rp2	5rsa	<i>1fgo</i>
1add	1azi	2fbj	2ptn	3cln	3sgb	5rub	<i>1qu3</i>
1adi	1crc	2fxb	2rhe	3cna	3tln	5tnc	<i>357d</i>
1ae4	1lzl	2gbp	2rn2	3dfr	4ape	6ldh	
1af3	1omd	2gcr	2rsp	3est	4dfr	6xia	
1agm	1rbp	2gn5	2sga	3fab	4fab	7api	
1agx	1rhd	2hfl	2sni	3gap	4fcn	7cat	
1ahn	1sgt	2hla	2ssi	3gbp	4hvp	7pcy	
1ai2	1tgn	2hmg	2stv	3gpd	4mdh	8adh	
1ak2	1ton	2lbp	2taa	3grs	4pep	8atc	

APPENDIX B Correlation Plots of all 153 Proteins & RNA (In alphabetical order)

APPENDIX C **Human and E. coli Source Proteins**

From the original 143-protein list, proteins of *Homo sapien* source and of *E. coli* source used for Figure 12 & 13

PDB ID	Protein Source	
	Homo sapien	Escherichia coli
	1a3s	1ab4
	1a6q	1adi
	1abm	1ahn
	1als	1ai2
	1lz1	1ako
	1rbp	1aln
	1ubq	1aob
	2cab	2gbp
	2fb4	2lbp
	2hla	2liv
	2pab	2rn2
	2plv	2trx
	2rhe	2tsc
	3fab	2wrp
	3gpd	3cla
	3grs	3gap
	3mcg	4dfr
	7api	8atc

APPENDIX D

Perl and Java Codes Used

```

#-----
# Perl Script
## Description: Extracts GI#, Score & total energy score
#               and puts it in db_import.csv
# Inputs: The input file must be called input.txt
# Outputs: db_import.csv contains the extracted data.
#-----
##### 1.)
# Open input file

open (INPUT, "input.txt");
open (DBFILE, ">db_import.csv"); # Import file for DB

##### 2.)
# Extract data &
# write to file
while (chop($line = <INPUT>)) {
    if($line =~ /GI# /) {
        print DBFILE $line;
        print DBFILE ",";
    }
    if($line =~ /the total energy score is /) {
        @temp = split(' ', $line); # extract number
        print DBFILE $temp[5];
        print DBFILE ",";
    }
    if($line =~ /Score = /) {
        @temp = split(' ', $line); # extract number
        print DBFILE $temp[2];
        print DBFILE "\n";
    }
}
print "Finished... (see file db_import.csv)";
close (INPUT);
close (DBFILE);

#-----
## Description: Extracts GI#, Identities (count)
#               and puts it in Identities.csv
# Inputs: The input file must be called input.txt
# Outputs: Identities.csv contains the extracted data.
#-----
##### 1.)
# Open input file
open (INPUT, "input.txt");
open (DBFILE, ">Identities.csv"); # Import file for DB
##### 2.)
# Extract data &
# write to file
while (chop($line = <INPUT>)) {
    if($line =~ /Identities = /) {
        @temp = split(' ', $line); # extract number

```

```

        @temp = split(/, $temp[2]);
        print DBFILE $temp[1];
        print DBFILE "\n";
    }
}
print "Finished... (see file Identities.csv)";
close (INPUT);
close (DBFILE);

#####
# Description:
# Counts the letters G, C, T, A, U
# in the file called ATGC.txt
#####
# Open Main file
open (INPUT, "ATGC.txt");
$G_count = 0;
$C_count = 0;
$T_count = 0;
$A_count = 0;
$U_count = 0;
while (!eof(INPUT) ) {
    $character = getc(INPUT);
    if($character == /G/) {
        $G_count++;
    }
    if($character == /C/) {
        $C_count++;
    }
    if($character == /T/) {
        $T_count++;
    }
    if($character == /A/) {
        $A_count++;
    }
    if($character == /U/) {
        $U_count++;
    }
} # end of while
$total = $G_count + $C_count + $T_count + $A_count + $U_count;
print "\n-----\n";
print " G count = ", $G_count, "\n";
print " C count = ", $C_count, "\n";
print " Total [G,C,T,A,U] = ", $total, "\n";
print "-----\n";
close (INPUT);

```

```

#####
# Description:
# Counts the letters GG, CC, TT, AA, UU
# in the file called ATGC.txt
#####
# Open Main file
open (INPUT, "ATGC.txt");
$GG_count = 0;
$CC_count = 0;
$TT_count = 0;

```

```

$AA_count = 0;
$UU_count = 0;
$XX_count = 0;
$global_count = 0;
$previous = getc(INPUT);
Count($previous);
$total = $GG_count + $CC_count + $TT_count + $AA_count + $UU_count + $XX_count;
print "\n-----\n";
print " GG count = ", $GG_count, "\n";
print " CC count = ", $CC_count, "\n";
print " TT count = ", $TT_count, "\n";
print " AA count = ", $AA_count, "\n";
print " UU count = ", $UU_count, "\n";
print " XX count = ", $XX_count, "\n";
#print " Global Cnt ", $global_count, "\n";
#print " Total [G,C,T,A,U] = ", $total, "\n";
print "-----\n";
close (INPUT);
sub Count {
    $prev = $_[0];
    $new = getc(INPUT);
    $global_count++;
    if(($new =~ /\x0a/) || ($new =~ /\x20/)) { # Skip newlines and spaces
        $new = getc(INPUT);
    }
    if($prev == $new) {
        if( $new == /G/) {
            $GG_count++;
            Count($new);
        }
        elsif( $new == /C/) {
            $CC_count++;
            Count($new);
        }
        elsif( $new == /T/) {
            $TT_count++;
            Count($new);
        }
        elsif( $new == /A/) {
            $AA_count++;
            Count($new);
        }
        elsif( $new == /U/) {
            $UU_count++;
            Count($new);
        }
        elsif( $new == /X/) {
            $XX_count++;
            Count($new);
        }
        else {
            print "Error: Only G, C, T, A, U, X allowed.\n";
            print "Check location: ", $global_count, "\n";
            print "char = ", $new, "\n";
        }
    }
}
elsif(eof(INPUT)) {
    print "EOF\n";
    return;
}

```

```

}
else {
    Count($new);
}
} # end Count

#####
# Description:
# Counts the letters AU, AT, UA, TA, GC
# in the file called ATGC.txt
#####
# Open Main file
open (INPUT, "ATGC.txt");
$AU_count = 0;
$AT_count = 0;
$UA_count = 0;
$TA_count = 0;
$GC_count = 0;
@array = ();
$i = 0;
# Remove all newlines & spaces #
while (!eof(INPUT) ) {
    $char = getc(INPUT);
    if(($char =~ /\x0a/) || ($char =~ /\x20/)) { # Skip newlines and spaces
        next;
    }
    else {
        $array[$i++] = $char;
    }
}
#print @array, "\n";
#print $i;
$j = 0;
while($j < $i) {
    if($array[$j] =~ /A/) {
        if($array[$j+1] =~ /U/) {
            $AU_count++;
        }
        if($array[$j+1] =~ /T/) {
            $AT_count++;
        }
    }
    elseif($array[$j] =~ /U/) {
        if($array[$j+1] =~ /A/) {
            $UA_count++;
        }
    }
    elseif($array[$j] =~ /T/) {
        if($array[$j+1] =~ /A/) {
            $TA_count++;
        }
    }
    elseif($array[$j] =~ /G/) {
        if($array[$j+1] =~ /C/) {
            $GC_count++;
        }
    }
    else {}
    $j++;
}

```

```

}
print "\n-----\n";
print " AU count = ", $AU_count, "\n";
print " AT count = ", $AT_count, "\n";
print " UA count = ", $UA_count, "\n";
print " TA count = ", $TA_count, "\n";
print " GC count = ", $GC_count, "\n";
#print " Global Cnt ", $global_count, "\n";
#print " Total [G,C,T,A,U] = ", $total, "\n";
print "\n";
close (INPUT);

```

JAVA CODE:

```

import java.io.*;
class FixBlast {
    void addToFile() {
        File inFile1 = new File("oldBlast.txt");
        String blast_file = new String();
        String last_gi = new String();
        String last_line = new String();
        try {
            BufferedReader in1 = new BufferedReader(new FileReader(inFile1));
            PrintWriter output = new PrintWriter(new BufferedWriter(
                (new FileWriter("blast.txt"))));
            while ((blast_file = in1.readLine()) != null) {
                if (blast_file.startsWith(">gi")) {
                    last_gi = blast_file;
                }
                if (blast_file.startsWith(">gi")
                    || blast_file.startsWith("Sbjct"))
                    last_line = blast_file;
                if (blast_file.startsWith(" Score")
                    && last_line.startsWith("Sbjct"))
                    output.println (last_gi);
                output.println(blast_file);
            }
            in1.close();
            output.close();
        } catch (FileNotFoundException e) {
            System.out.println ("File not found " + e);
        } catch (IOException e) {
            System.out.println("problem " + e);
        }
    }
    public static void main(String[] args) {
        FixBlast lastFile = new FixBlast();
        lastFile.addToFile();
    }
}
import java.io.*;
import java.util.*;
public class ParseFile {
    private PrintWriter output;
    private static String output_file_name;
    private static boolean complement = false;
    void openOutputFile() {
        File outputFile = new File(output_file_name);
        try {

```

```

        output = new PrintWriter(new BufferedWriter
                                   (new FileWriter(outputFile)));
    } catch (Exception e) {};
}
void closeOutputFile() {
    output.println();
    output.close();
}
void addToFile (double sequ_num) {
    addToFile("\nGI# " + sequ_num + "\n");
}
void addToFile(String sequ) {
    FileWriter output_file;
    if (complement) {
        sequ = sequ.replace('a','x');
        sequ = sequ.replace('t','a');
        sequ = sequ.replace('x','t');
        sequ = sequ.replace('g','x');
        sequ = sequ.replace('c','g');
        sequ = sequ.replace('x','c');
    }
    output.print(sequ);
}

void parseSequence(String q_type) {
    StreamTokenizer input_stream;
    openOutputFile();
    try {
        input_stream = new StreamTokenizer (new BufferedReader
                                             (new FileReader("blast.txt")));
        //input_stream.wordChars('-', '-');
        while (input_stream.nextToken() != StreamTokenizer.TT_EOF)
            switch (input_stream.ttype) {
                case StreamTokenizer.TT_WORD:
                    if (input_stream.sval.equals("gi")) {
                        while (input_stream.nextToken() !=
                               StreamTokenizer.TT_NUMBER);
                        addToFile(input_stream.nval);
                    } else if (input_stream.sval.equals(q_type)) {
                        // XXX fix this - problem
                        while (input_stream.nextToken() != StreamTokenizer.TT_WORD)
                            if (input_stream.ttype == '-')
                                output.print('-');
                        addToFile(input_stream.sval);
                    }
                    break;
                default: // nothing
            }
    } catch (Exception e) {};
    closeOutputFile();
}

public static void main(String[] args) {
    String s_squ = new String();
    ParseFile seq = new ParseFile();
    output_file_name = "blastQueryOutput.txt";
    seq.parseSequence("Query");
    output_file_name = "blastSbjctOutput.txt";
    complement = true;
    seq.parseSequence("Sbjct");
}

```



```

    }
}
import java.io.*;
import java.util.*;
class ERNA {
    private static PrintWriter output;
    private static int total_match;
    private static int total_gap;
    ERNA() {
        File outputFile = new File("FinalOutput.txt");
        try {
            output = new PrintWriter(new BufferedWriter
                (new FileWriter(outputFile)));
        } catch (Exception e) {} ;
    }
    double[] unmatched_int_eng = {0.0,0.8,1.3,1.7,2.1,2.5,
        2.6,2.8,3.1,3.6,0.0,4.4,0.0,5.1,0.0,5.1,0.0,5.6,0.0};
    double[] gap_int_eng = {3.3,5.2,6.0,6.7,7.4,8.2,9.1,
        10.0,10.5,11.0,11.8,12.5,13.0,13.6,14};
    private HashMap hm = new HashMap();
    // set hashtable
    void setupHashMap () {
        hm.put("aatt",new Double(-0.9));
        hm.put("ttaa",new Double(-0.9));
        hm.put("atta",new Double(-0.9));
        hm.put("taat",new Double(-1.1));
        hm.put("cagt",new Double(-1.8));
        hm.put("tgac",new Double(-1.8));
        hm.put("ctga",new Double(-1.7));
        hm.put("agtc",new Double(-1.7));
        hm.put("gact",new Double(-2.3));
        hm.put("tcag",new Double(-2.3));
        hm.put("gtca",new Double(-2.1));
        hm.put("actg",new Double(-2.1));
        hm.put("cggc",new Double(-2.0));
        hm.put("gccg",new Double(-3.4));
        hm.put("ggcc",new Double(-2.9));
        hm.put("ccgg",new Double(-2.9));
    }
    void getSequence() {
        File inputFile1 = new File("blastQueryOutput.txt");
        File inputFile2 = new File("blastSbjctOutput.txt");
        String input_qry = new String();
        String input_sbj = new String();
        try {
            BufferedReader in1 = new BufferedReader (new FileReader(inputFile1));
            BufferedReader in2 = new BufferedReader (new FileReader(inputFile2));

            int counter = 0;
            while (((input_qry = in1.readLine()) != null)
                && ((input_sbj = in2.readLine()) != null)) {
                counter++;
                // skip blank lines
                if(input_qry.length() <= 3 && input_sbj.length() <= 3)
                {
                    //System.out.println("S.L.");
                    continue;
                }
                if (input_qry.length() != input_sbj.length()) {

```

```

        System.out.println ("BIG PROBLEM");
        System.exit(0);
    }
    if (input_qry.startsWith("GI"))
        if (input_qry.equals(input_sbj))
            output.print("\n\n\n" + input_qry);
        else
            output.print("\n\n\n" + input_qry);
    else
    {
        //System.out.println(counter);
        //System.out.println(" ");
        processInput(input_qry, input_sbj);
        output.flush();
    }
} // end while
System.out.println(counter);
in1.close();
in2.close();
//inputFile1.delete();
//inputFile2.delete();
} catch (FileNotFoundException e) {
    System.out.println ("File not found " + e);
} catch (IOException e) {
    System.out.println("problem " + e);
}
}

void processInput(String qry_input, String sbj_input) {
    double[] weight = new double[qry_input.length()];
    calThermoEng(qry_input, sbj_input, weight);
    printOutput(qry_input, sbj_input, weight);
}

boolean geneticMatch(char gene1, char gene2) {
    boolean match = false;
    int value = gene1 + gene2; // addition is associative
    if ((value == 213) || (value == 202))
        match = true;
    return match;
}

void calThermoEng(String sequ1, String sequ2, double[] weight) {
    boolean prev_match;
    double score = 0, gap_pen = 0, unmatch_pen = 0;
    int unmatch_count = 0, gap_count = 0;
    total_match = 0;
    total_gap = 0;
    if (geneticMatch(sequ1.charAt(0), sequ2.charAt(0))) {
        prev_match = true;
        weight[0] = 0.0;
        total_match++;
    } else {
        prev_match = false;
        weight[0] = 0.8;
    }
    score += weight[0];
    for (int i = 1; i < sequ1.length(); i++) {
        if (geneticMatch(sequ1.charAt(i), sequ2.charAt(i))) {
            if (prev_match) {
                String ashs = sequ1.substring(i-1, i+1)
                    + sequ2.substring(i-1, i+1);

```

```

        Double ash = (Double)(hm.get(ashes));
        weight[i] = ash.doubleValue();
        score += weight[i];
    } else
        weight[i] = 0.0;
    prev_match = true;
    unmatched_count = 0;
    gap_count = 0;
    total_match++;
} else if (sequ1.charAt(i) == '.' || sequ2.charAt(i) == '.') {
    gap_count++;
    total_gap++;
    if (gap_count > 1)
        weight[i - 1] = 0.0;
    gap_pen = gap_int_eng[gap_count - 1];
    weight[i] = gap_pen;
    prev_match = false;
} else {
    unmatched_count += 2;
    unmatched_pen = unmatched_int_eng[unmatched_count - 1];
    weight[i] = unmatched_pen;
    prev_match = false;
    gap_count = 0;
}
}
}

void printEngVal(double[] weight, int cur_pos, int length) {
    for(int i = cur_pos; i < cur_pos + length; i++) {
        if (weight[i] == 0.0) {
            output.print(" ");
        }
        else {
            output.print(weight[i] < 0.0 ? " " : " ");
            output.print(weight[i]);
        }
    }
}

void printGeneVal (String gene_sequ) {
    output.print(" ");
    for (int i=0; i < gene_sequ.length(); i++) {
        output.print(gene_sequ.charAt(i));
        output.print(" ");
    }
}

void printOutput(String qry_str, String sbj_str, double[] weight) {
    int cur_pos = 0, length = 0;
    double total_eng = 0;
    int total_length = qry_str.length();
    while (cur_pos < total_length) {
        if (total_length - cur_pos <= 14) {
            length = total_length - cur_pos;
        }
        else
            length = 14;
        output.print("\nQuery: ");
        printGeneVal(qry_str.substring(cur_pos, cur_pos + length));
        output.print("\nEnerg: ");
        printEngVal(weight, cur_pos, length);
    }
}

```

```

        output.println("\nSbjct: ");
        printGeneVal(sbj_str.substring(cur_pos, cur_pos + length));
        cur_pos += length;
        output.println();
    }
    for (int i = 0; i < weight.length; i++)
        total_eng += weight[i];
    output.println("the total energy score is " + total_eng);
    output.print("The number of matchers: " + total_match + " / " +
        qry_str.length());
    output.println(", Total gaps " + total_gap);
}

public static void main(String[] args) {
    ERNA seq = new ERNA();
    seq.setupHashMap();

    try {
        seq.getSequence();
    }
    catch(Exception e)
    {
        System.out.println("—");
    }
    output.close();
}
}

import java.io.*;
class FixFinal {
    void addToFile() {
        File outputFile = new File("last.txt");
        File inputFile1 = new File("FinalOutput.txt");
        File inputFile2 = new File("blast.txt");
        String blast_file = new String();
        String eng_file = new String();
        try {
            BufferedReader in1 = new BufferedReader (new FileReader(inputFile1));
            BufferedReader in2 = new BufferedReader (new FileReader(inputFile2));
            PrintWriter output = new PrintWriter(new BufferedWriter
                (new FileWriter(outputFile)));
            while ((eng_file = in1.readLine()) != null) {
                if (eng_file.startsWith("GI")) {
                    output.println (eng_file);
                    do {
                        eng_file = in1.readLine();
                        output.println (eng_file);
                    } while (!eng_file.startsWith("The"));
                    output.println ("nBLAST Results ");
                    do {
                        blast_file = in2.readLine();
                    } while (!blast_file.startsWith(">gi"));
                    do {
                        output.println(blast_file);
                        blast_file = in2.readLine();
                    } while (!blast_file.startsWith("Query"));
                    output.println ("n\n NEXT RESULT \n\n");
                }
            }
        }
        /*if (input_qry.startsWith("GI"))
            if (input_qry.equals(input_sbj))
                output.print("\n\n\n" + input_qry);
        */
    }
}

```

```

        else
            output.print("\n\n\n" + input_qry);
        else
            processInput(input_qry, input_sbj);
        }*/
        in1.close();
        in2.close();
        output.close();
    } catch (FileNotFoundException e) {
        System.out.println ("File not found " + e);
    } catch (IOException e) {
        System.out.println("problem " + e);
    }
}

static void main(String[] args) {
    FixFinal lastFile = new FixFinal();
    lastFile.addToFile();
}
}

```