JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-10 E-ISSN: 2347-2693

Minimum Free Energy-Based Amino Acid Sequence Permutation From Amino Acid

E. Lloyd-Yemoh[1]*, H.B. Shi[2]

[1*] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

**Corresponding Author: elias_lloyd@live.com, Tel.: +8615651725575*

Available online at: www.ijcseonline.org

Received: 17/Sep/2017, Revised: 26/Sep/2017, Accepted: 16/Oct/2017, Published: 30/Oct/2017

Abstract**—** Computationally speaking, there are a few ways to tackle problems of rule-based permutations and iterations. This paper seeks to explore two algorithms and their possible application in the field of bioinformatics and biochemical engineering. We believe that accurately predicting RNA secondary structure formations can only be achieved by extensive analysis of specific RNA folds that have already been documented to occur in nature and others like them that have the same Amino acid sequence structure and similar minimal free energies. This paper focuses on algorithms to extract every single RNA sequence that fits a given amino acid sequence. We concern ourselves mainly with the computation intensive issue of the outputting various permutations of given protein sequences and their respective minimal free energies.

Results: We present a way to computationally improve analysis of secondary structure minimization. Using C++, Sequence permutations of amino acids are extracted to be analyzed in terms of minimum free energies. ViennaRNA-2.1.6 is used to facilitate our computation of the RNA fold and the corresponding minimal free energy. The Odometer Weighted Counter (OWC) approach comes in second with its critical length of six amino acids and a computations time of 68 seconds. The Vector Permutation Mapping (VPM) approach comes in as the more desirable approach with a critical length of 10, and a computation time of 26896 seconds. All tests were made on critical path length of sequences.

An output of importance to our paper is the minimal free energy of each RNA sequence that the ViennaRNA RNAfold function processes. Analysis of the resulting minimal free energies in comparison to already documented RNA strings in nature is the key to more effective secondary structure prediction.

Keywords**—**RNA, minimal free energy, amino acid, folding, ViennaRNA.

I. Introduction

Deoxyribonucleic acid (DNA) is a molecule that holds all genetic instructions that are necessary for an organism's specific growth and development. The genetic information found in DNA is what is responsible for how organisms look, function and propagate. It is usually found in nature as double strands winding in the shape of a double helix. DNA is perhaps the most important molecular structure in cells because it holds all information about the organism. In protein synthesis it is too risky to have DNA travel outside to active sites to instruct on how proteins should be synthesized. Damage to DNA will cause issues in later generations and avoiding that risk is what brings us to RNA [1]. RNA stands for Ribonucleic acid and is a major contributing agent in coding, decoding and translation of genes. It is largely responsible for decoding and transporting genetic information to facilitate the synthesis of proteins at targeted areas [1]. RNA is made up of nucleotides that are assembled in a form a chain that we will henceforth be referring to as its sequence. Below, we see a sample RNA sequence from [2] which folds into the graphical representation seen in Fig 1.RNA is usually found in nature to be folded onto itself instead of bonding with other strands like DNA does. Understanding and accurately predicting how RNA folds on to itself (secondary structure) is the key to major breakthroughs in medicine. Many viruses encode their genetic information in RNA structures therefore unlocking the secrets behind RNA folding and protein synthesis will mean a way to better understand viral micro-organisms and how to manipulate and treat them [3,4]. Naturally occurring RNA folds are not easily predicted but there is a method that is used to measure the possibility [5].

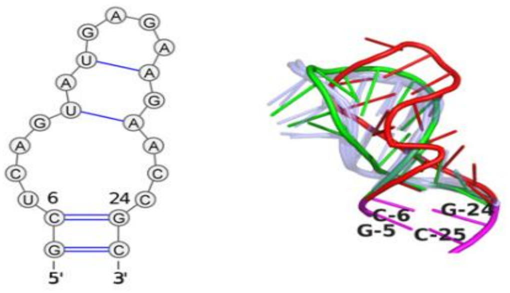

Fig. 1. Sample RNA secondary structure folded from GCUCAGUAUGAGAAGAACCGC [2]

The likelihood of a particular RNA occurring in nature is termed its minimal free energy. It makes it possible to quantify and therefore measure how stable a particular RNA secondary structure is [6]. That in itself does not guarantee that said sequence will occur in nature. Ideally, being able to accurately predict protein formation and RNA folding will help scientists to make big advancements in biochemical engineering. Scientist have been studying DNA and RNA sequences for years, trying to find the next big breakthrough or a way to fully understand how they occur in the hopes of one day being able to predict with a higher level of accuracy, their formation and occurrence in nature even hopefully manipulate it. The key to unlocking these secrets of nature we believe lies in the minimal free energy of the RNA sequence and its fold [7]. Every RNA sequence can be represented in terms of amino acids and this means, with the number of possible codon replacements for each amino acid, an amino acid sequence can be rewritten in many different ways that still fall under the amino sequence rule.

It is our belief that can only be achieved by extensive analysis of specific RNA folds that have already been documented to occur in nature and others like them that have the same Amino acid sequence structure and similar minimal free energies.

Table 1 Odometer Weighted Counter Pseudo Code

Algorithm Pseudo Code

```
Create library of Amino acid objects;
Create object array and store Amino objects in;
Accept string input of amino acid sequence;
Create dynamic object array;
for(i=0;i¡input.length;i++)
( Load dynamic array corresponding object from input;)
Call function leftCall;
return;
```

Predicting a secondary structure can also be made more accurate by comparing structures common to multiple sequences. The accuracy of structure prediction is significantly improved by predicting a secondary structure common to multiple sequences.

We propose that secondary structure prediction accuracy can be improved by amino acid permutation analysis. Given an amino acid sequence (preferably of a RNA sequence with a documented or confirmed secondary structure), analysis of its secondary structure in comparison to other similar structures will uncover patterns. Even more accurate, will be structures that fall under same categorized amino acids and have similar MFE. Either by exclusion or inclusion, the study of similar sequences will help scientists better understand and predict folding more accurately. This theory may further be improved by the application of folding kinetics. In this paper, we focus on determination by free energy minimization. This cannot be done without making available every RNA sequence the fits the amino acid rule. This paper tackles the computation intensive aspect of extracting the many sequence possibilities. [7]

Section I contains the introduction of biological terminology and background , Section II contain the related work of Jun et al. and Hajiaghayi et al, Section III contains two algorithms developed to solve our permutation problem, Section IV contains the results and discussion of experiments carried out on each of the algorithms and Section V concludes research work with future directions.

II. Related Work

Jun et al. in their attempt to develop a method to adjust the expression level of certain proteins without making a change to the amino acid sequence, made a few strides in the computation we seek to improve [8]. Theirs is somewhat limiting in the sense that to optimize the algorithm for the particular task at hand, they removed certain codons they deemed less preferred or not often used. After this, every possible combination of the sequence of amino acids of a targeted protein is calculated. We do not seek to optimize our algorithm in the same way because its our belief that to accurately predict protein synthesis and offer a truly complete library for analysis, every codon in the amino acid sequence (more frequently and less frequently used alike) has to be present. With such computationally intense tasks like this every resource counts. A lot would be dependent on the kind of CPU and accompanying memory that is present [9]. Commonly used free energy minimization tools for predicting RNA folds are based on dynamic programming algorithms. These algorithms are basically able to analyze all possible secondary structures for a given sequence without actually generating structures [10, 11].

In [11], M. Hajiaghyi recognized the need for larger datasets in the pursuit of more reliable measures of RNA structure prediction in algorithms. Caution is required when using average accuracies on certain classes due to the limited size of available datasets. Datasets like S-Full with a compilation of 3245 RNA sequences, MT with S Ribosomal RNA, Group I intron, Group II intron, Ribonuclease P RNA, Signal Recognition RNA and Transfer RNA and lastly, MA a subset of the S-Full that are in the RNA classes included in MT. In our experiments, we were able to generate databases of over 32GB worth of RNA sequences in raw data. The equivalent of millions of RNA strings [11].

III. METHODOLOGY

Our aim is to return every possible permutation that fits the rule of a given amino acid sequence. The goal is to find the RNA strings that are most likely to occur in nature. This is achieved by running every output that matches the amino acid sequence through Vienna's RNAfold function [12] to determine the minimum free energy of each sequence.

Odometer Weighted Counter

The idea behind the approach is to start from the smallest weighted unit (the rightmost unit) and work our way gradually to the leftmost unit after exhausting every possible permutation to the right. Starting from the leftmost unit, we apply weights in descending order. Just like how centimeters run into meters which then run into kilometers and so on. The idea behind it is such that the smaller the weight of the units the more flags for execution they get and they end up having to make more turns than those with higher weights. In figure 3, we can tell how the lower-weighted units or unit wheels have to be exhausted before their immediate left unit can make a step. At any point in time, the visible area, the area easily seen refers to the current value as far as the sequence is concerned.

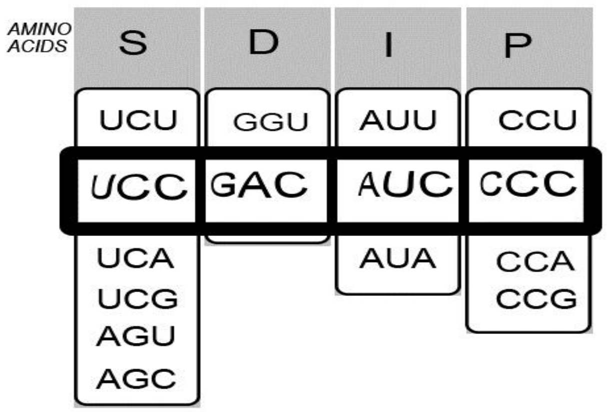

Fig. 2. Implementation of the Odometer concept

A recursive function is a function that calls itself in its implementation. In most cases it's a function that calls itself however many times needed to complete a set task. Recursive calls usually require a base condition, an exit if you may, so as not to cause a crush or go on endlessly. In this algorithm we use two partially recursive calls. They are so called because they are not like your usual recursive function. Instead of repetitively calling themselves over and over, they occasionally (depending on what conditions are met) call each other [13]. These two functions, leftCall and rightCall work hand in hand to act as one large recursive call. On one hand the rightCall function takes in 4 arguments which include a pointer to an array of objects, pointer to a specific object of a class, the size input and a pointer to the counter for current object flagged active.

A typical call of rightCall would result in a few checks to seek a go ahead for execution after which all temp values of each amino acid object in the array will be printed. In the event that all codons of the particular object are exhausted leftCall will be called. The function leftCall on the other hand after making its increment, flags the last object in the array for execution unless its codons has been exhausted in which case it calls itself on the immediate left object.

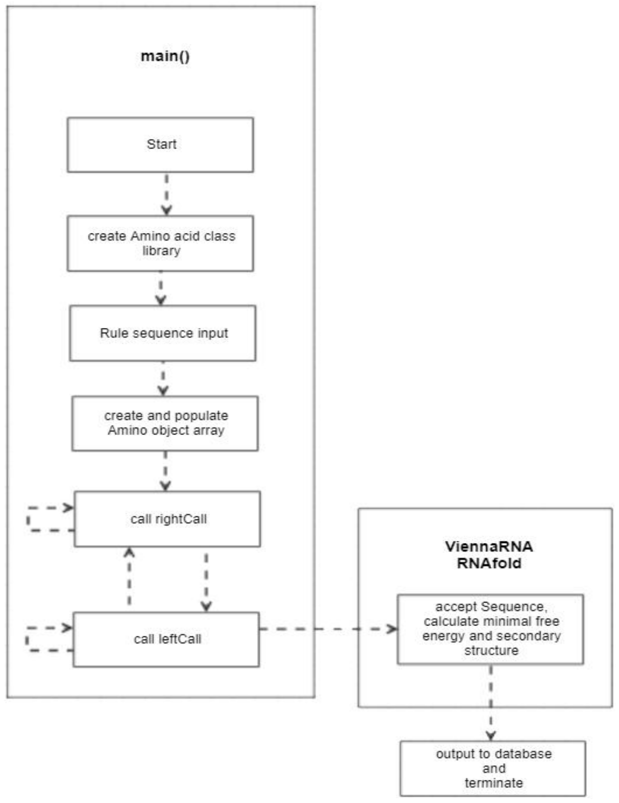
main()
Start
create Amino acid class library
Rule sequence input
create and populate Amino object array
call rightCall
call leftCall
ViennaRNA RNAfold
accept Sequence, calculate minimal free energy and secondary structure
output to database and terminate

Fig. 3. Diagram showing program workflow

The task at hand is computationally complex and resource consuming. There are concerns of various complications and limitations (relative to the type of hardware present in the system) when it comes to memory usage due to the large number of function calls associated with long input sequences or even short sequences that contain a high amino acids with the max number of codons.

Using this approach we placed the amino acid classes in an object array in the order of the sequence rule. We load the array with class objects much like how the wheels are loaded onto the beam in fig 3. Each amino acid object contains all the necessary information about itself that help in tracking and querying. Each object contains: name of said amino acid, the number of codons it houses and each codon name.

Applying that to the odometer concept, we place the array of codons of the respective amino acids on the unit spaces allocated to each amino acid. The idea is to make each unit moves one step only after the object at its immediate right has exhausted every codon it contains (exhausted here mean it has been displayed). This approach ensures that we find the permutations in an organized and orderly manner and when the leftmost unit has exhausted all its codons, the program terminates.

Not all amino acids have the same number of codons however they all range from one to six codons per amino acid so number of steps or rotations may vary. So the number of rotations vary relative to the number of codons of the current unit (amino acid) and number of cycles or revolutions depend on the number of units to the left of the current unit. The equation below sums up the expression and definition of number of revolutions (ΔR) required at any index on the amino object array to reach the terminal state of the program.

$$\Delta R = ILC \tag{1}$$

By definition, the number of revolutions is the product of the number of codons of every amino acid with a higher weight than the current unit. This is represented by ILC. The function ILC like factorial also requires a base case at index 0.

$$ILC = (IL)_{num\_of\_codons} \times ILC; \tag{2}$$

$$ILC[0] = 1; \tag{3}$$

Vector Permutation Mapping

This algorithm is based on a depth weighted recursion system. A typical permutation problem like this is tackled by C++ by exhaustively searching through the depth of the vector starting from the leftmost element in the vector. Given a sample vector {A, B, C], weights are allocated to each vector element according to their frequency. If A's weight is 3, it implies that it appears three times in the sequence, Applying that, the C++11 built in algorithm maps and produces all possibly permutation of a specified rule, in this case an amino acid sequence

Table 2. Vector Weighted Permutation Pseudo Code

Vector Weighted Permutation Pseudo Code

Create a map library for amino acids and respective codons;
For(outputString: translate(input) mapping)
{
Temp=”./RNAfold” + outputString;
System(temp);
}
End of program.

Underneath, the vector permutation takes an amino acid vector and in the same concept of applying weights, the different codons are applied to each and then the mapping takes care of the rest.

Due to the nature of the algorithm, every possible scenario has to be considered, anticipated and coded. It is important to note that the function-call loop can only be terminated by the leftCall function. The algorithm is such that only two conditions have to be fulfilled to terminate, first that the amino object flagged as 'current' is the first in the array and second that it exhausted its codons.

By way of Optimization, a couple of major modifications were made to improve the overall efficiency of the program. In preliminary tests, all functions were making copies of and referring to the copies of amino acids. This caused problems with a build-up of large amounts of variables in the memory and an inefficient way of tracking program progress. The use of pointers later proved to be invaluable because changes were made directly to the original copies.

The use of a 2-dimensional array to store the amino acid library and populate the requested string proved to be very limiting because if led to the use of multiple depth nested loops that created unnecessary overheads. This led to the necessity to use another data structure in its place. Classes and objects turned out to be the solution since a class hold different types of data in one structural container.

IV. RESULTS AND DISCUSSION

Experimentation and testing involved the use of what we termed critical lengths of Amino acid sequences. This means that only amino acids with the maximum number of six codons were used in our test amino acid sequences. This way test results are more standard and prudent.

It is important to note however that a sample critical length string “RLS” will output 216 different sequences which is equivalent the number of output permutations of amino string "MIFAIWI" or longer if we keep adding amino acids with only one codon. It's a form of compressed standard of amino acid expression. It therefore performed exceptionally well in comparison to Vector Permutation Mapping (VPM) and the original C++ implementation of the Odometer Weighted counter (OWC). All tests were made on critical length sequences.

All experiments were made on an Intel Core i7-6700CPU@3.4GHz x 8 processor, running Ubuntu 15.10(64-bit) with 32GB RAM.

Overall, it is quite visible that the VWP algorithm is the superior algorithm in this comparison. In terms of speed and computation power, it comes in ahead of the OWC algorithm [13]

Table 3. Odometer Weighted Counter Vector and Permutation Mapping Comparison

		ELAPSED TIME			
		OWC		VWP	
Amino Seq	length	Real (s)	CPU (t)	Real (s)	CPU (t)
”RLS”	3	0		0	
”RLSR”	4	2	0.068247	2	0.068774
”RLSRL”	5	11	0.472538	11	0.433519
”RLSRLS”	6	68	3.459838	66	4.353482
”RLSRLSR”	7	N/A	N/A	498	67.696404
”RLSRLSRL”	8	N/A	N/A	5978	1788.71

Storage of resulting information was an issue that we still struggle to improve on daily. Resulting information comes in three main forms, the RNA sequence, the minimal free energy of respective string and the graphical representation of the secondary structure of the fold. Even though the graphical fold structure was not our priority, we still sought to save it for review. After various tests, first with .txt file extension and then with .csv, it was determined that the sheer size of outputs required large amounts of storage. Storage of the graphical representation was halted to make way for more sequences to be stored. With the switch from .txt to database storage and the storage of only critical information, the storage issue was overcome.

V. CONCLUSION

Using CPU and available memory resources alone in computing rule-based permutation is difficult and very resource consuming. The algorithms themselves have a few limitations that cause segmentation faults and core dumps.

We have succeeded in producing the various permutation of amino acids in the form of RNA sequences and their corresponding minimal free energies. A quick query of the database would show the lowest of the minimum free energies and also that of the base RNA sequence. Analysis of these similar RNA chains even those that have lower minimum free energies than the documented natural occurring RNA chains will help make secondary structure prediction more accurate.

The amount of processing power required for 3D rendering of new games and this has led to the production of very powerful GPUs with high computational potential. Until recently the computation potential outside the normal purpose of these GPUs has not been tapped.

We plan to use OpenCL in future computations to calculate permutation iterations on Graphic Processing units (GPUs) and to eventually introduce FPGA's to the computation. At the time of writing this paper, the OpenCL implementation of the OWC algorithm was under test on AMD GPUs.

OpenCL stands for Open computing language and as the name implies, it's a cross-platform, royalty-free programming language standard. It is a programming language that runs on CPU's, GPUs, FPGA's and other like processors [14,15] Field Programmable Gate Arrays are able to be programmed to match optimal schematics for a specific computations. It is our belief that by using FPGA boards we can cut out unwanted pathways and optimize the algorithm and its implementation in a more targeted approach

ACKNOWLEDGMENT

This research was supported by the College of computer Science, Nanjing University of Aeronautics and Astronautics, Mr. and Mrs. Lloyd-Yemoh and the expertise of Justice Adeenze-Kangah (senior programmer and database expert).

REFERENCES

[1] RF Gesteland, TR Cech, JF Atkins, "*The RNA World, edn 3*", Cold Spring Harbor Laboratory Press, 2005.

[2] J. Li, J. Zhang, J. Wang, W Li, Wang , "*Structure Prediction of RNA Loops with a Probabilistic Approach*", PLoS Comput Biol 12(8): e1005032. doi:10.1371/journal.pcbi.1005032 N,2016

[3] G.M Cooper; R.E Hausman. "*The Cell A Molecular Approach (3rd ed.)",* Sinauer. pp. 261–76, 297, 339–44. ISBN 0-87893-214-3,2004

[4] H Lodish, A Berk, S L Zipursky, P. Matsudaira, D. Baltimore, J,. Darnell, "*Molecular Cell Biology. 4th edition*".W. H. Freeman; 2000.

[5] J. S. Mattick; M. J. Gagen, "*The evolution of controlled multitasked gene networks: the role of introns and other noncoding RNAs in the development of complex organisms*", Mol. Biol. Evol. 18(9): 1611–30. doi: 10.1093 / oxfordjournals.molbev.a003951, September, 2001

[6] D. H. Mathews, "*Predicting a set of minimal free energy RNA secondary structures common to two sequences*", Vol. 21 no. 10, pages 2246–2253, 2005.

[7] S. Chen, "*RNA folding: conformational statistics, folding kinetics, and ion electrostatics*", Annu. Rev. Biophys. 37:199–214, 2008

[8] J. Yin, H. Tian, L.Bao, X. Dai. X. Gao, W.Yao, *"An alternative method of enhancing the expression level of heterologous protein in Escherichia coli*",Biochemical and Biophysical Research Communications 455 (2014) 198‑204, 2014

[9] S. R Eddy: "*How do RNA folding algorithms work?",* Nat Biotechnol 22:1457-1458,2004

[10] D. H. Mathews, M Zuker, "*Predictive methods using RNA sequences in Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*", edn 3. Edited by Baxevenis A, Oullette F, John.Wiley and Sons:143-170, 2004

[11] Hajiaghayi et al., "*Analysis of energy-based algorithms for RNA secondary structure prediction*". BMC Bioinformatics 13:22, 2012

[12] R. Lorenz, S. H. Bernhart, C. H. Siederdissen, H. Tafer, C. Flamm, P.F. Stadler,I.L. Hofacker, "*ViennaRNA Package 2.0, Algorithms*" Mol. Biol. 6, 2011

[13] J. S. Mattick. "*The hidden genetic program of complex organisms*". Scientific American. 291 (4): 60–7. PMID 15487671. doi:10.1038/scientificamerican1004-60, October 2004

[14] Khronos OpenCL Working Group,Aaftab Munshi."*The OpenCL Specification*".Version: 1.2. Revision: 19,2011

[15] Matthew Scarpino, "*OpenCL in Action*", Manning Publications ISBN 9781617290176,2012

Authors Profile

Mr. E. Lloyd-Yemoh ,a Ghanaian national, pursed Bachelor of Engineering in Science and Technology (Software Engineering and Management) from Nanjing University of Aeronotics and Astronautics in year 2013. He is currently pursuing a Masters in Computer Science and Engineering at the prior mentioned institution. His main research work focuses on Algorithm development and Optimization by leveraging parallel computing. He has 3 years of Research Experience.

Mr Huibin Shi obtained Master of Software Engineering and PhD in Compter Science respectively in year 2002 and 2006 from the University of York, UK. He is currently working as Associate Professor in College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China since 2009. He has co-published more than 20 research papers in reputed domestic and international journals and conferences. His main research work focuses on GPU/FPGA Accelerated Computing Bioinformatics, Cryptography Algorithms, Big Data Analytics, Reliable Embedded System, Dynamic Partial Reconfigurable System, Computer Architecture, IoT and Computational Intelligence based Intelligent manufacturing, Compiling technology.