Exhaustive Algorithm Approach to tRNA Secondary Structure Prediction

Steven Li[1, a)] and Bridget Foley[1, b)]
San Jose State University[c)]

(Dated: November 13, 2019)

We attempted to create an algorithm that tackles the problem of finding the secondary structure of a tRNA by finding the 4 regions within tRNA that corresponds to how the overall tRNA secondary structure will form. The focus of the algorithm is to the find the 4 pairs between the forward and backward direction tRNA sequence that would be representative of the 4 folding portion of the tRNA then give that information to a biologist so that can inspect and infer what specific structure those respective folding portion will churn out. From that maybe even predict what the overall secondary structure will look like and what function corresponds to which structure configuration or base sequencing. We found that this algorithm works as intended in finding the 4 pairs that corresponds to each folding region but only works in the ideal case that each base in the pairing sequence is identically complimentary to the opposition which is not the case in reality because some pairings are strong enough to pull together adjacent bases that don't have an identical complimentary to fold.

I. MOTIVATION

A tRNA's function is mainly derived from how it's secondary structure folds out, the secondary structure has 4 main folding regions: Aminoacyl stem, T-arm, Anticodon Arm and the D-Arm that determines the overall structure depicted in **Figure 1** that is very similar to a clover leaf shape. Predicting the location of where these 4 folding regions are within the tRNA sequence can give insight to what overall secondary structure will be constructed. The 4 folding regions can be predicted by finding specific pairs in the tRNA where the forward direction and backward direction sequence form a complimentary pair with each other.

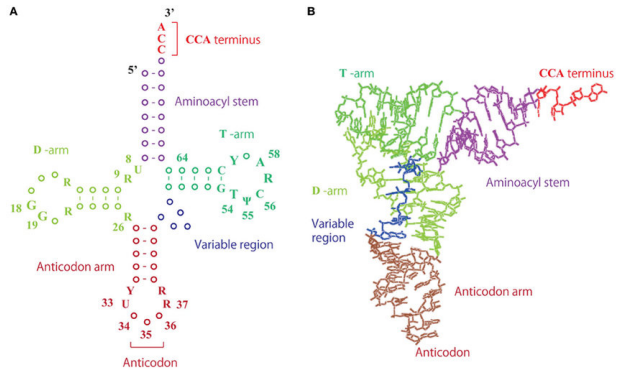

Figure 1. The structure of a tRNA where there are 4 folding regions; Aminoacyl stem, T-arm, Anti-codon Arm and the D-arm. Along with visual representation of how the overall secondary structure of a tRNA can look like.

[a)] Electronic mail: steven.li@sjsu.edu
[b)] Electronic mail: bridget.foley@sjsu.edu
[c)] Also at San Jose State University.

II. METHODOLOGY

The methodology used to tackle the problem of finding the secondary structure of tRNA was to use a brute-force or exhaustive algorithm that goes through every possible iteration of pairings in the forward and backward direction of a tRNA sequence to find appropriate and efficient pairings for the 4 folding regions: aminoacyl stem, T-arm, Anticodon Arm and the D-arm. **Figure 2.** shows a simplified flowchart of the algorithm implemented and used in an attempt to predict the 4 folding regions within the tRNA that is then used for further analysis to predict the tRNA's secondary structure and therefore it's important functions.

A. Aminoacyl Stem

The first portion of the algorithm is to find the Aminoacyl Stem portion of the tRNA which is a pair of 6 or 7 bases that are usually complimentary with each other but that's not essentially necessarily. The algorithm takes all possible iterations of 6 and 7 base sequences in both the forward and backward direction of the tRNA, the last n-th iteration in the forward direction is not taken because it would just be a duplicate of the first n-th sequence in the backward direction. The forward and backward iteration of each n-th sequence length are put into their own separate array that is within another array that denotes what the n-th sequence length is and the forward and backward iterations are compared with each other to see if they are complimentary identical to each other. After a pairing is deemed complimentary identical by the algorithm, it will also calculate a score parameter that is calculated depending on the distance the two sequences are from each other by (1).

$$score = |index_{>>} - index_{<<}| \tag{1}$$

This is done by array indexing, if a match is found

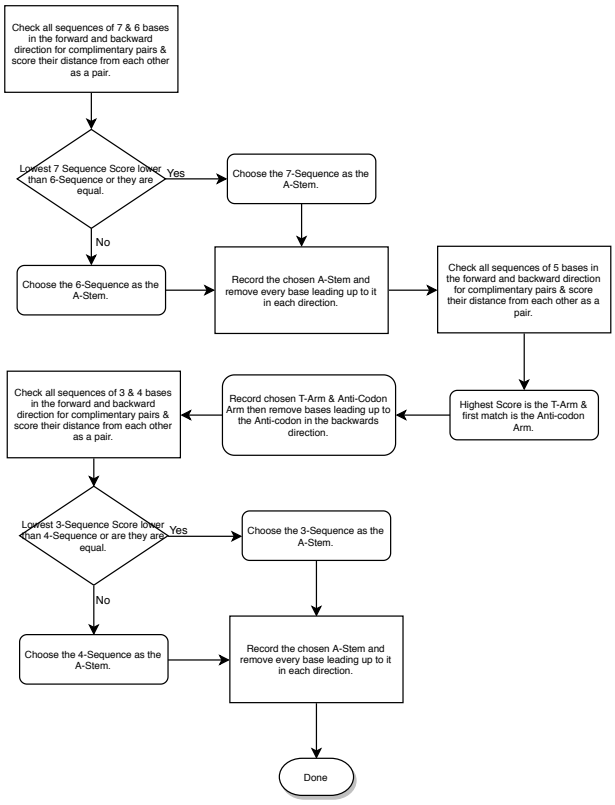

Figure 2. Algorithm flowchart

at the very start of the sequence such that the first n-th sequence in both the forward and backward direction are found to be complimentary identical, the scoring will be zero. After all the pairings have been found and the scores have been calculated, if either n-th sequence pairing that was not found such as the array of pairings is empty, the array is replaced with 'empty' in both forward and backward direction and the scoring is set to 100 because usually tRNA sequences never exceed 100 bases making it the worst score possible.

After this is done, all that is left is to determine which pairing is the most appropriate and most probable Aminoacyl Stem. This is done by finding the minimum score between both n-th sequence pairings, if the minimum is the same for both n-th sequences then the greater sequence length is assumed to be the most probable Aminoacyl Stem. Otherwise, the lowest score is considered to be the most probable Aminoacyl Stem because they are the farthest apart from each other and this fold usually wants to maximize the distinct from each sequence to it's complimentary.

B. T-Arm & Anti-codon Arm

After finding the Aminoacyl Stem of the tRNA, we remove it from the tRNA sequence such as all the bases leading up to both the forward and backward sequence are removed to filter out bases that are already used and the ones that are no longer useful. The algorithm is then repeated again similar to the Aminoacyl Stem but instead of checking for both 6 and 7 length sequences, we only exhaustively search for all the 5 sequence long pairings because the T-Arm & Anti-Codon Arm are both 5 sequences long. After all the pairings and scores have been calculated, the highest scoring pair is the most probable T-Arm because the first sequence should be very close the second sequence because the T-Arm is located near the end of the forward direction tRNA and the beginning of the backward direction tRNA reading. While the most probable Anti-Codon Arm is the first pairing that was found by the algorithm.

C. D-Arm

Continuing what was done for the T-Arm & Anti-codon Arm, we remove those two pairing sequences from the tRNA sequence to filter out used and useless bases leaving only the tRNA sequence that is open to be used to make the last region of the secondary structure, the D-Arm. The D-Arm can be 3 or 4 bases in length requiring us to use exhaustive search on both 3 and 4 length sequences for the remaining tRNA but for the D-Arm, we use a modified scoring formula (2) based on experimental testing that seemed to churn out D-Arm that were correct based on experimental data that is on ***Genomic tRNA Database, GtRNAdb***.

$$score = index_{>>} - index_{<<} \tag{2}$$

Out of the pairs that are found, we determine the most probable D-Arm based on the scoring parameter. If the lowest score of the 4 base sequence is smaller than the lowest score of the 3 base sequence, then we take the lowest scoring 4 base sequence as the D-Arm. If both n-th sequence scoring are the same then we still take the 4 base sequence as the D-Arm because it is more probable for that extra base to pair up then not. Otherwise, the 3 base sequence is taken as the D-Arm.

D. Tool Creation

With the creation of the algorithm explained above, instead of just creating a program that utilizes the algorithm we decided that creating a web application utilizing the algorithm would be much more user friendly, easily accessible and much more convenient than requiring users install external tools and struggle to get them working. Because of this decision, I had decided to write the program in the programming language PHP as the backend and Materialize as a front-end framework alongside using a web server solution stack package called XAMPP to create a web application hosted on AWS using EC2 with

a Ubuntu operating system. This web application is very light weight, easily accessible and all work is done on the server-side. It was originally intended to equip the web application with a database so that input searches were recorded and easily retrieved when identical inputs were queried but that seemed unnecessary because the server-side computation was not as computationally intensive as originally thought because tRNA sequences usually do not exceed 100 sequences.

Instructions to use the tool is to go http://bit.ly/tRNApredict where the display should be identical to **Figure 3.** Then input your desired tRNA sequence and submit the query.

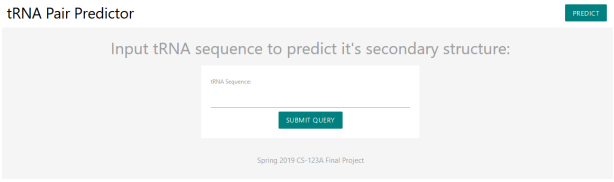

Figure 3. Web Application Query Page

The web application should quickly redirect the user to a new page that shows the results where there are 5 containers. The containers contain: input tRNA, Aminoacyl Stem pair, T-Arm pair, Anti-Codon Arm pair and the D-arm pair in that order shown in **Figure 4.**

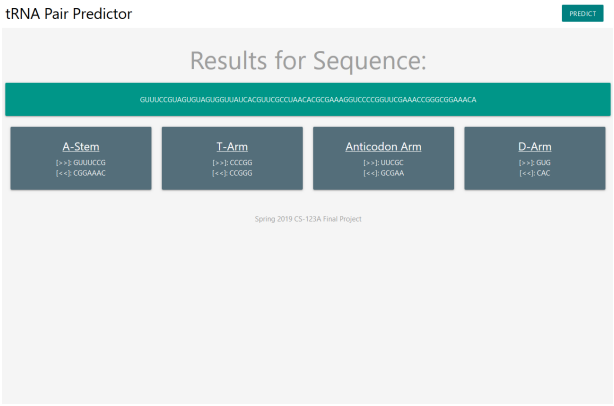

Figure 4. Web Application Query Results Page

III. ANALYSIS

The sequences of tRNA that were used to test the program were from the RSCB Protein Data Bank[1],[2], as well as GtRNA Data Bank from UC Santa Cruz.[3] The tRNA sourced from the Protein Data Bank had been tested and confirmed experimentally, and the crystal structure was confirmed through other methods besides a computer algorithm. Sequences tested from this database include yeast tRNA that codes for phenylalanine, and yeast tRNA that codes for aspartic acid. The sequence had to be pulled from the .pdb file, and initially the matching bases had to be paired up manually by comparing the sequence string to the 3D file in Chimera.[4] After manually confirming the Aminoacyl Stem, T-Arm, Anti-Codon, and D-arm, the sequence could be put in our algorithm.

To maintain a similar sequence structure for the program, many of the tRNA sequences tested came from the GtRNA Data Bank. All of these sequences were able to be input in our algorithm without misalignment. This Data Bank also has a tRNA sequence predictor, called tRNAscan-SE. We compared our algorithm output to this predictor as well, just to make sure they had similar results. Some tRNA sequences tested from this site include tRNA that codes for textitE. coli cysteine[5], *Drosophila melanogaster* (House fly) alanine[6], *Ceratotherium simum* (white rhinocerous) glutamic acid[7], and *Mus musculus* (House mouse) valine[8]. Each of these sequences' output matched the tRNAscan-SE output each time, except once where there was a documented wobble base on the house mouse valine.

The sequences that came from the protein data bank may have varied slightly in their length, which is why there were some problems inputting the sequence into the algorithm. Modifying the start site of the sequence allowed it to go smoothly. Each of the sequences from the GtRNA Data Bank were input smoothly, and gave similar results.

IV. DISCUSSION

This algorithm recognizes the four standard RNA bases: adenine, cytosine, guanine, and uracil. These are the most common nitrogenous bases found in tRNA and are widely present in many tRNA sequences. This algorithm has demonstrated its ability to not only accurately match complementary bases, but also identify the different parts of the secondary structure of tRNA. If this comprehensive algorithm were to be modified in the future, there are updates that could be made to increase the scope of the tRNA sequences it could identify. There are modified bases found in some tRNA sequences that can hinder the pairing parameters. Modified nucleotides are any nucleotides that contain an additional functional group on their initial structure, and that have a different code than the standard four bases.[9] Some examples include methylguanine, dihydrouracil, and xanthine. These numerous modified bases have different pairing rules with the common bases, so future updates to this algorithm could include the ability to recognize their unique activity and how they contribute to the secondary structure of some tRNA molecules.

Another update that could be applied to this program is recognition of wobble base pairs. This is related to the program's recognition of modified base terminology, but also makes the program more accurate. Wobble base pairing is also called non-Watson-Crick base pairing, and

it is most common in tRNA, which contributes to why an update on this would be beneficial.[10] In these arrangements, modified bases bond to the common bases, but they do not follow standard Watson-Crick rules either. Guanine, for instance, may also bind to Uracil. These pairings are important for tRNA because it allows for specificity without an overwhelming number of tRNA species.

Recognizing these additions to the program will not only make it more accurate, but with expand our tRNA species scope. There are tRNA strands that may follow Watson-Crick pairing completely, but in order to use this program on more species, these updates will give the algorithm the scope it needs to work on more tRNA.

REFERENCES

[1] B. R. J. A. Hingerty, B.E., (1978), 10.2210/pdb2TRA/pdb.

[2] D. P. M. D. Westhof, E., (1987), 10.2210/pdb2TRA/pdb.

[3] T. Chan, P.P. Lowe, Nucl. Acids Res. (2016), 10.1093/nar/gkv1309.

[4] H. C. C. G. G. D. M. E. F. T. Pettersen EF, Goddard TD, J Comput Chem. (2004), 10.1002/jcc.20084.

[5] http://gtrnadb.ucsc.edu/genomes/bacteria/Delf$_{C}s1_{4}/genes/tRNACys-GCA-1-1.html$, $.http://gtrnadb.ucsc.edu/genomes/eukaryota/Dmela6/genes/tRNAAla-AGC-2-10.html$, .

[6] http://gtrnadb.ucsc.edu/genomes/eukaryota/Csimu1/genes/JH767733.trna11-GluCTC.html, .

[8] http://gtrnadb.ucsc.edu/Mmusc10/genes/tRNA Val-AAC-1-2.html, .

[9] M. M. Christian Lorenz, Christina E. Lünse, Biomolecules **7**, 35 (2017).

[10] H.-M. Eun, Methods in Enzymology (1996), 10.1016/B978-0-12-243740-3.X5000-5.