Benchmarking Current RNA Folding Software and Improvements to the Current Regime

Jianglin Liu, Bingxi Li, and Gregory Rehm

University of California, Davis {jiliu, bxli, grehm}@ucdavis.edu

November 26, 2015

Abstract

Finding the secondary structure of RNA is important for understanding how RNA will interact in a cell. Frequently computational algorithms are used to determine structure due to difficulties extracting good *in vivo* data of RNA structures. Many of the algorithms for RNA folding are computationally complex. In this paper we establish benchmarks for two commonly used RNA folding packages, mfold, and RNAfold Vienna, and compare them to an improved serial and parallel Four-Russians folding algorithm. We show *RNAfold Vienna* is faster than mfold in evaluating RNA of both linear and circular varieties. We also show that these software packages would benefit greatly from using new improved RNA folding algorithms especially for strands of RNA greater than 1000 nucleobases. These results will help software maintainers to understand the benefit of updating their algorithms. The results will also help guide users when choosing RNA folding software when looking for the most computationally optimal package.

1 Introduction

RNA is an essential macromolecule used in protein formation and performs other essential functions within the body(4). RNA does not stay in single stranded form and instead folds on itself to create the lowest energy conformation possible to ensure thermodynamic stability(6). When folding, RNA forms a 2D secondary structure(5) with A matching to U and G to C (figure 1). Using this data we can find a 3D tertiary structure(5).

Our paper focuses on benchmarking 2D secondary structure prediction software based off the Nussinov dynamic programming algorithm(3) for RNA folding. This algorithm is $O(n^3)$ time complexity. There have been multiple attempts to parallelize the Nussinov algorithm(15; 16) which have resulted in large speed increases, however, CPU intensive algorithms only slightly lowered the bound up until 2010(12; 13). In 2010 the Nussinov bound that was significantly improved by the Frid-Gusfield Four Russians method which established you could perform the DP method in $O(\frac{n^3}{log(n)})$ time(1). Later a parallel method

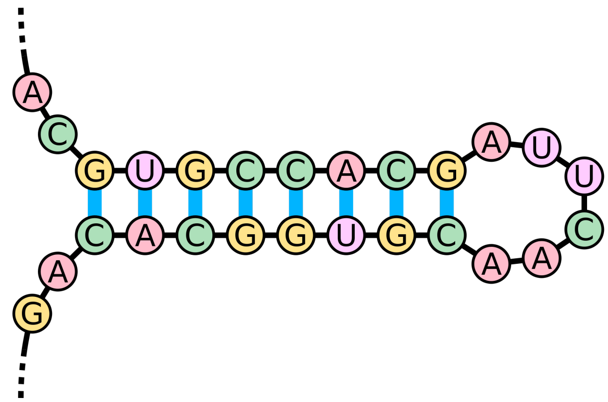

Figure 1: Example of an RNA molecule folding

of the Four Russians algorithm presented proof that you can lower this bound to $O(\frac{n^2}{log(n)})$ inside an NVIDIA CUDA environment(2).

There are two major RNA folding software packages, mfold(8; 9), and the Vienna RNA Package(7). These both utilize the Nussimov method to return results of the RNA secondary structure by finding the lowest possible thermodynamic conformations of the RNA(9; 7). In order to tell which was faster we performed application level benchmarks(14) to see which of these two applications could more quickly fold RNA of the circular and linear variety. The way mfold is written allows linear RNA to be treated as exceptional variants of circular ones(17). RNAfold Vienna was initially optimized to only handle linear RNA(17) but later improvements enabled it to speed the folding of circular ones(17). As a result we wished to determine what kind of speed difference still exists between mfold and Vienna when performing analysis on circular RNA. To see how these folding applications could benefit from replacing their use of the Nussinov method with newer ones we then performed micro-benchmarks(10) of the Nussinov, the serial Frid-Gusfield method, and the parallel Four Russians method.

Our main finding when performing application benchmarks on mfold and Vienna was that Vienna is significantly faster than mfold for linear and cicular RNA. For micro-benchmarks we found that the Frid Gusfield method returned results much faster than the Nussinov method especially for larger strands of RNA. However the parallel Four Russians CUDA algorithm returned results significantly quicker than the other serial methods. The results of the micro-benchmarks show that both mfold and RNAfold Vienna could experience significant speed increases if they implemented the Frid-Gusfield method. Furthermore the authors of this paper would recommend both software packages to support GPU hardware to achieve even greater speed gains when inside a parallel capable environment.

2 Methods

2.1 Standardizing the Testing Environment

Benchmarking is renown as a difficult thing to perform effectively(10; 14). There are many processes that can be executing on a computer at any one moment that it is possible that a benchmark can give inaccurate information due to conflicting processes running in the background(10). As a result we used a machine solely dedicated for benchmarking and no other tasks. We also standardized on the following specifications for our runs(11):

Architecture	**Operating System**	**Compiler**
8 core Intel i7 CPU 4.00 GHz 16G RAM GeForce GTX 960	Linux 4.2.5-201 Fedora 22	GCC 5.1.1-4.fc22
”	”	gcc-gfortran 5.1.1-4.fc22
”	”	NVIDIA CUDA version 5.5

We used the following applications with corresponding versions and requirements in our test runs:

Application	**Version**	**Requirements**
mfold	3.6	GCC, Fortran
RNAfold Vienna	2.1.9	GCC
Nussinov	N/A	GCC
Frid-Gusfield Four Russians	N/A	GCC
Parallel Four Russians	N/A	GCC, NVIDIA CUDA

Our testing architecture was laid out where we would SSH into the benchmarking machine and then execute tests. Test results would then be reported back to the user's central machine where they could be stored in a database for later analysis (Figure 2). Our testing required no internet connectivity besides the ssh access required to initiate our testing so all calculations were performed locally. Also there were no IO operations except for post processing of mfold and Vienna results.

2.2 Data Inputs

For input data we give inputs of RNA as strings in a file. An example of this would be the 10 character RNA string AUGCCAUGGA. This same RNA sequence can be treated as circular by providing parameters to the mfold and RNAfold Vienna programs that tell it the RNA is circular(18; 19). We vary the inputs of RNA by length in this paper but not by sequence. The reasoning for this is because the algorithms we are benchmarking should be more dependant on the length of input for their time benchmarks rather than the the exact sequence used. The exact sequences of RNA are included in supplementary material.

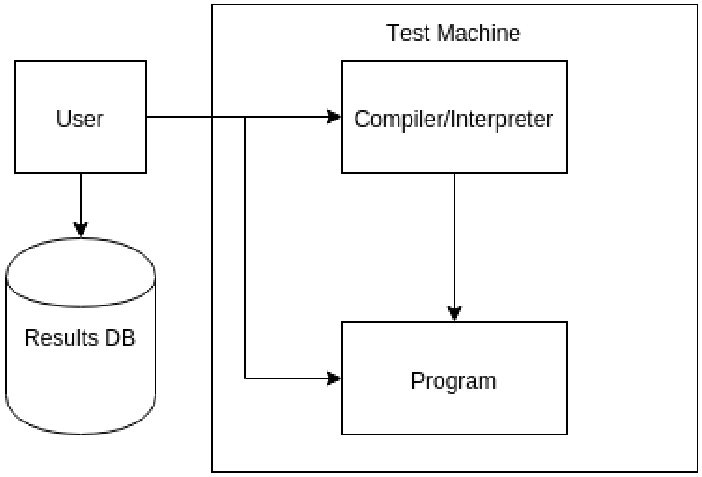

Figure 2: Test Architecture

2.3 Application Benchmarks

The first type of benchmark we perform is the application level benchmark. An application benchmark is designed to measure the performance of an entire application and the resources it consumes on an individual machine(20). In our case we wish to evaluate the amount of time that mfold and RNAfold Vienna take to return RNA secondary structures given different length RNA strands varying on linear and circular variety. Since a single run of an application may vary in time even for identical inputs. Because of this we evaluate each input of RNA 30 times and report the mean μ, standard deviation, σ, of the runs corresponding to each sequence length.

2.4 Micro-benchmarks

The most basic type of benchmark to perform is the micro-benchmark. The micro-benchmark is a single piece of code executed many times in serial so that we can get a profile of its run characteristics(14; 10). Once these characteristics are observed we can then make inferences about its performance and ways that it can be improved.

Micro-benchmarks have the downside of losing generality of performance across the entire application(14; 10). A good example of this is if an IO heavy function made many consecutive calls to the *read* function on the OS while the rest of the application made no calls to *read* whatsoever. If we tried to generalize this one function to the rest of the application we would misguidedly attempt to optimize disk IO across our entire system.

We avoid this trap in our paper by benchmarking a stand-alone implementation of the Nussinov algorithm. We then report these results back to our test results database for later analysis. After this we compare these results to runs of the serial Frid-Gusfield algorithm and parallel Frid-Gusfield algorithm.

2.5 Frid-Gusfield Four-Russians Algorithm

The Four-Russians Algorithm (1) is an algorithm to improve the above-mentioned Nussinov $O(n^3)$ Algorithm by Four-Russian method. The Frid and Gusfield is an $O(\frac{n^3}{\log n})$ algorithm. The Four Russian algorithm achieves this speed up by understanding that we can make certain optimizations to the matrix of matching base pairs required by the Nussinov algorithm. Particularly, the values along a column from bottom to top and along a row from left to right are monotonically non-decreasing. Consecutive cells differ at most by 1(1). As a result we can perform pre-processing of specific operations that the Nussinov algorithm must compute manually.

2.6 CUDA Parallel Implementation for F-G Method

Compute Unified Device Architecture (CUDA) is a parallel computing platform created by NVIDIA. By using CUDA API, Venkatachalam presented an $O(\frac{n^2}{\log n})$ algorithm for RNA folding is presented (2). The CUDA implementation parallelizes the two-vector method so that achieve an enhancement of another factor of $O(n)$.

3 Results

3.1 Application Benchmarks

In this part, we performed the benchmark for two packages using the Dynamic Programing (DP) paradigm with both linear and circular RNA. We report the number of RNA bases or **size**, the number of times the folding program was run with our specific input or **N runs**, the mean timing of the runs as $\boldsymbol{\mu}$, and the standard deviation of the runs denoted as $\boldsymbol{\sigma}$. For the benchmarking of both linear and circular types of RNA, we select the RNA of the size, ranging from 200 nucleobases to 1100 nucleobases. The times to fold these structures are compared in the following Table 1.

Timing of *m-fold* package for linear RNA (sec.)										
size	200	300	400	500	600	700	800	900	1000	1100
N runs	30	30	30	30	30	30	30	30	30	30
μ	2.523	6.635	6.313	8.336	10.492	9.384	12.212	14.319	14.402	21.496
σ	0.343	0.251	0.180	0.547	0.144	0.094	0.176	0.118	0.211	0.75
Timing of $Vienna$ for linear RNA (sec.)										
size	200	300	400	500	600	700	800	900	1000	1100
N runs	30	30	30	30	30	30	30	30	30	30
μ	0.029	0.072	0.112	0.163	0.508	0.491	1.934	11.723	8.296	7.585
σ	0.006	0.001	0.001	0.001	0.002	0.002	0.006	0.336	0.011	0.131

Table 1: The time taken by two packages to predict secondary structures of linear RNA of different length.

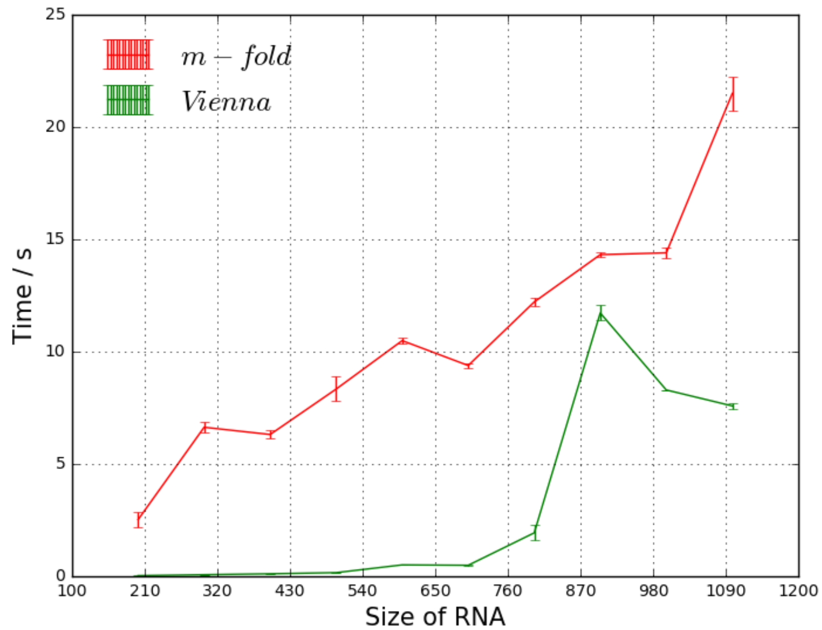

Figure 3: Benchmark of *Vienna* and *m-fold* packages with linear RNA of different sizes.

We can see from Figure 3. that *m-fold* takes the most time to complete. However as mentioned before, Vienna is optimized to handle linear RNA so these results are not necessarily surprising. We benchmark circular RNA. The execution time to determine their secondary structures are listed in the following Table 2.

Timing of *m-fold* package for circular RNA (sec.)										
size	200	300	400	500	600	700	800	900	1000	1100
N runs	30	30	30	30	30	30	30	30	30	30
μ	2.409	6.633	6.269	8.889	11.274	9.074	11.358	14.678	16.712	21.765
σ	0.060	0.354	0.362	0.159	0.1176	0.161	0.364	0.1999	0.285	0.424
Timing of *Vienna* for circular RNA (sec.)										
size	200	300	400	500	600	700	800	900	1000	1100
N runs	30	30	30	30	30	30	30	30	30	30
μ	0.027	0.084	0.129	0.153	0.664	0.351	1.943	7.915	50.15	13.667
σ	0.002	0.001	0.001	0.002	0.002	0.002	0.006	0.036	0.126	0.18

Table 2: The time taken by two packages to predict secondary structures of circular RNA of different length.

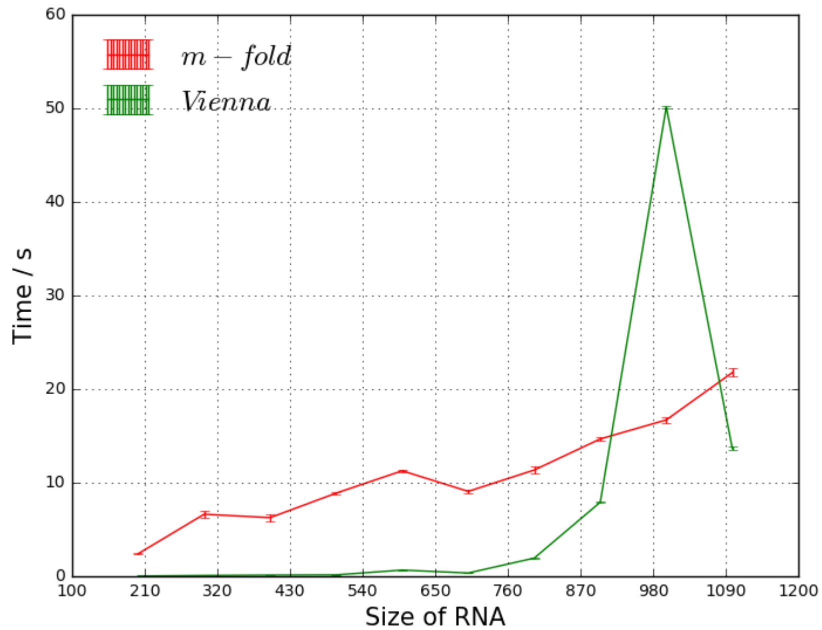

Figure 4: Benchmark of *Vienna* and *m-fold* with circular RNA of different sizes.

As we can see from Figure 4 Vienna initially outperforms *m-fold* but as RNA size hits 1000 bases the time to complete the folding increases dramatically. However for 1100 bases Vienna once again outperforms *mfold*. We will discuss this finding in our discussion section.

3.2 Micro-benchmarks

The timing for Nussinov, Frid-Gusfield (FG), and CUDA Four-Russian algorithms are listed below with different sizes of RNA sequences 500 to 6000. The following times are averages after 5 runs each. The data is presented in Table 3.

Timing of Algorithms for RNA Folding (sec.)							
size	500	1000	2000	3000	4000	5000	6000
N runs	5	5	5	5	5	5	5
Nussinov	0.2790	2.0751	16.7033	57.8146	145.2998	301.4874	519.6531
F-G	0.0903	0.6092	5.5868	19.6117	49.3309	95.6461	162.9072
CUDA	0.0088	0.1988	0.4690	1.1943	2.5817	4.8506	8.2735

Table 3: The time take by *Nussinov*, *F-G* and *CUDA* to construct the secondary structures of RNA of different size.

By plotting Figure 5, we observe the *F-G* method has a vast advantage over *Nussinov* when the size of RNA sequence is larger than 3000, and the *CUDA* method becomes much more useful around this point as well.

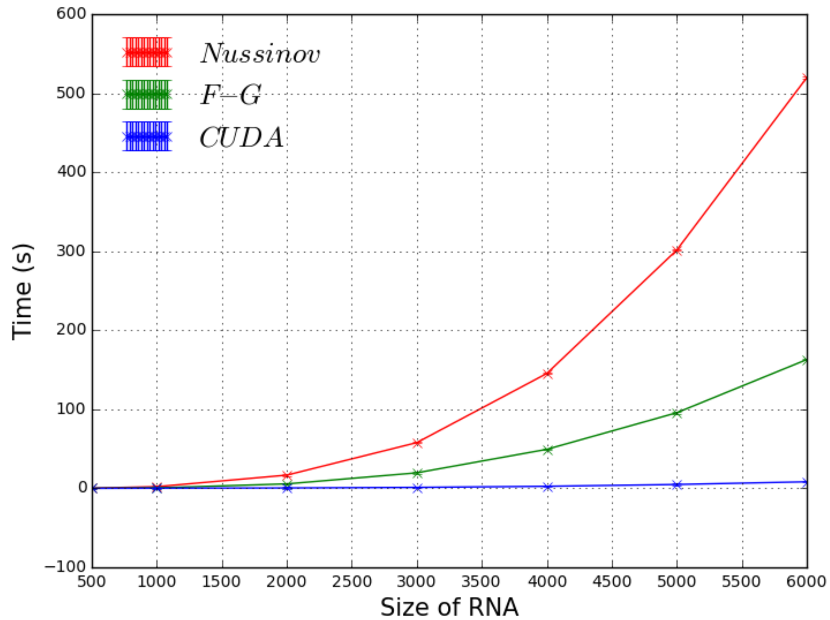

Figure 5: Timing of Algorithms for RNA Folding (sec.)

4 Conclusion and Discussion

In this paper, we performed application benchmarks for *m-fold* and *RNAfold Vienna* and micro-benchmarks for the Nussinov, Frid-Gusfield, and parallel Four Russians algorithms. For the application benchmark, *RNAfold Vienna* and *m-fold* have been applied to predict the secondary structure of both linear and circular RNA. In both linear and circular cases, the *Vienna* package is clearly optimal for almost all chains of RNA of both linear and circular types. While *m-fold* is time optimal in the folding of our single circular RNA strand of 1000 nucleobases. We believe this to be an abberation brought on by differences in the ways that *mfold* and *Vienna* both handle the evaluations of their thermodynamic models. We conclude that while on a whole Vienna will be faster than mfold, there exist strands of RNA that mfold is more adept at handling especially ones of the circular type given mfold is more properly architected to handle circular RNA.

When we come to the micro-benchmarks, we found that the *Nussinov* DP method becomes appreciably slower, particularly to the human user, in the cases that the number of nucleotides in the RNA sequence is greater than 1000. This is intrinsically caused by the fact the algorithm is of $O(n^3)$ time complexity. The *Frid-Gusfield* method (1) has enhanced the performance by a divisive factor of $log(n)$. In cases where RNA strands are greater than 2000 nucleobases however, its speed still ranges from 5 to 19 seconds. With the introduction of *CUDA* the two-vector method invented by Venkatachalam(2), we achieve a much closer to linearly increasing time complexity for different sizes of RNA sequences. Thus when dealing with super long RNA sequences of greater than 2000 nucleobases investing in the use of parallel algorithms and CUDA becomes much more cost effective. This is true especially if we consider that we would be waiting 519 seconds just to run an RNA sequence of 6000 bases through the Nussinov method and 162 seconds using Frid-Gusfield.

For future work we wish to benchmark *RNAfold Vienna* and *mfold* for RNA sequences greater than 1100 nucleobases. We have some evidence now that *m-fold* is superior to Vienna when predicting secondary structures of certain circular RNA but we do not know for what strands of RNA this holds nor for what length. Later work will let us know which types of RNA strands *mfold* outperforms *Vienna* and why the performance between the two applications differs for circular RNA. For our experiment we only used a single RNA sequence for each length but future experiments can involve multiple random RNA sequences of the same length. This way we can obtain a more interesting standard deviation of timing compared to what we have found in this paper. We can apply the same methodology to our micro-benchmarks as well. Although results may not vary as significantly because the thermodynamic rules that *Vienna* and *m-fold* employ are not present.

We believe that future work on integrating the Frid-Gusfield and parallel CUDA methods into *m-fold* and *Vienna* will be very important. RNA packages like *RNAfold Vienna* and *m-fold* must also take into account complex thermodynamic rules in order to find both optimal and suboptimal RNA structures. These caluculations all use the Nussinov method and as a result slow execution time down even more in an application setting. As a result implementation of an improved folding algorithm like Frid-Gusfield or the parallel Four Russians introduced by Venkatachalam is imperative especially as we discover more, and longer chains of RNA that are biologically significant.

5 Author Contributions

Bingxi Li contributed the idea for the paper, running RNAfold Vienna and providing benchmarking results. Jianglin Liu set up CUDA environment, repaired the source code for microbenchmark algorithms, and ran them and provided results. Gregory Rehm ran the mfold application benchmarks, performed background research on both RNA folding software and benchmarking, and provided the run methodology for our experiments.

References

[1] Frid Y, Gusfield D. *A simple, practical and complete $O(n^3)$-time algorithm for RNA folding using the Four-Russians Speedup.* Algorithms Mol Biol 2010, 5:13

[2] Venkatachalam B, Gusfield D. *Faster algorithms for RNA-folding using the Four-Russians method.* Algorithms for Molecular Biology20149:5.

[3] Nussinov R, Jacobson A. *Fast algorithm for predicting the secondary structure of single-stranded RNA*. Proc. Nati. Acad. Sci. USA Vol. 77, No. 11, pp. 6309-6313, November 1980.

[4] Mathews D, Turner D. *Prediction of RNA secondary structure by free energy minimization.* Current Opinion in Structural Biology 2006, 16:270–278.

[5] McCaskill J.S. *The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure*. Biopolymers, Vol. 29,1105-1119 (1990)

[6] Herschlag D. *RNA Chaperones and the RNA Folding Problem*. Vol. 270, No. 36, Issue of September 8, pp. 20871–20874, 1995

[7] Hofacker I. L, Fontana W, Stadler P. F, Bonhoeffer L. S, Tacker M, Schuster P. *Fast Folding and Comparison of RNA Secondary Structures*. Monatshefte ftir Chemie 125, 167-188 (1994)

[8] Zuker M. *Computer Prediction of RNA Structure*. Methods in Enzymology, vol. 180.

[9] Zuker M, Steigler P. *Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information*. Nucleic Acids Research vol. 9 Number 11981.

[10] Gregg B. *Systems Performance; Enterprise and the Cloud*. Pearson Education 2014 Upper Saddle River, NJ

[11] Altun, O. *Clustering Application Benchmark*. IISWC.2006.302742

[12] Akutsu, T. *Approximation and Exact Algorithms for RNA Secondary Structure Prediction and Recognition of Stochastic Context-free Languages* Journal of Combinatorial Optimization 3, 321–336 (1999)

[13] Chan TM. *More Algorithms for All-Pairs Shortest Paths in Weighted Graphs*. SIAM J Comput 2010, 39(5):2075-2089

[14] Cantrill B. *Eulogy for a benchmark*. The Observation Deck Web. http://dtrace.org/blogs/bmc/2009/02/02/eulogy-for-a-benchmark/. 2009.

[15] Rizk G, Lavenier D. *GPU accelerated Rna folding algorithm*. Allen, G.; Nabrzyski, J.; Seidel, E.; Albada, G.D. van; Dongarra, J.; Sloot, P.M.A. 9th International Conference on Computational Science, May 2009, Baton Rouge, United States. Springer., 5544, pp.1031, 2009, LNCS. ¡10.1000.ISBN: 978-3-642-01969-2¿. ¡hal-00425543¿

[16] Chang D, Kimmer C, Ming O. *Accelerating the Nussinov RNA Folding Algorithm with CUDA/GPU* Signal Processing and Information Technology (ISSPIT), 2010 IEEE International Symposium on, 15-18 Dec. 2010, pp. 120-125

[17] Hofacker I, Stadler P. *Memory efficient folding algorithms for circular RNA secondary structures*. Bioinformatics (2006) 22 (10): 1172-1176.

[18] Zuker M, Matthews D.H, Turner D.H. *Algorithms and Thermodynamics for RNA Secondary Structure Prediction: a Practical Guide*. The mfold Web Server Web. http://unafold.rna.albany.edu/doc/mfold-manual/mfold-3.0-manual.pdf.gz.

[19] Hofacker I, Fontana W, Bonhoeffer S, Stadler P.F, Lorenz R. *RNAFOLD*. Theoretical Biochemistry Group Institute for Theoretical Chemistry Web. https://www.tbi.univie.ac.at/RNA/RNAfold.1.html#heading5

[20] Jain, R. *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*. Wiley Computer Publishing, John Wiley & Sons, Inc. ISBN: 0471503363 Pub Date: 05/01/91.