

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 24, 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 Four-Russians folding algorithm. We show ... 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

Vienna.

These results show us that ... for natively differentiating between mfold and Vienna. They also help explain that both pieces of software can 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
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.

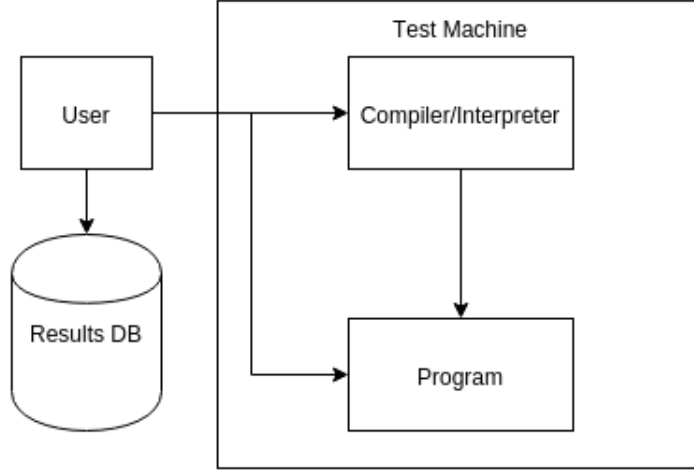


Figure 2: Test Architecture

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).

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^2 , 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 only parts of the code that execute the Nussinov algorithm in mfold and the Vienna package. 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.

The Four-Russians Algorithm (1) is a two-vector algorithm to improve the above-mentioned Nussinov $O(n^3)$ Algorithm. 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, in Venkatachalam B (2), an $O(\frac{n^2}{\log n})$ algorithm for RNA folding is presented.

The CUDA implementation parallelizes the two-vector method so that achieve an enhancement of another factor of $O(\log n)$ (2).

3 Results

The timing for Nussinov, FG and CUDA Four-Russian algorithms are listed below with different sizes of RNA sequences.

Timing of Algorithms for RNA Folding (sec.)							
Size	500	1000	2000	3000	4000	5000	6000
Nussinov	0.2790	2.0751	16.7033	57.8146	145.2998	301.4874	519.6531
FG	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

The data are plotted visually as below:

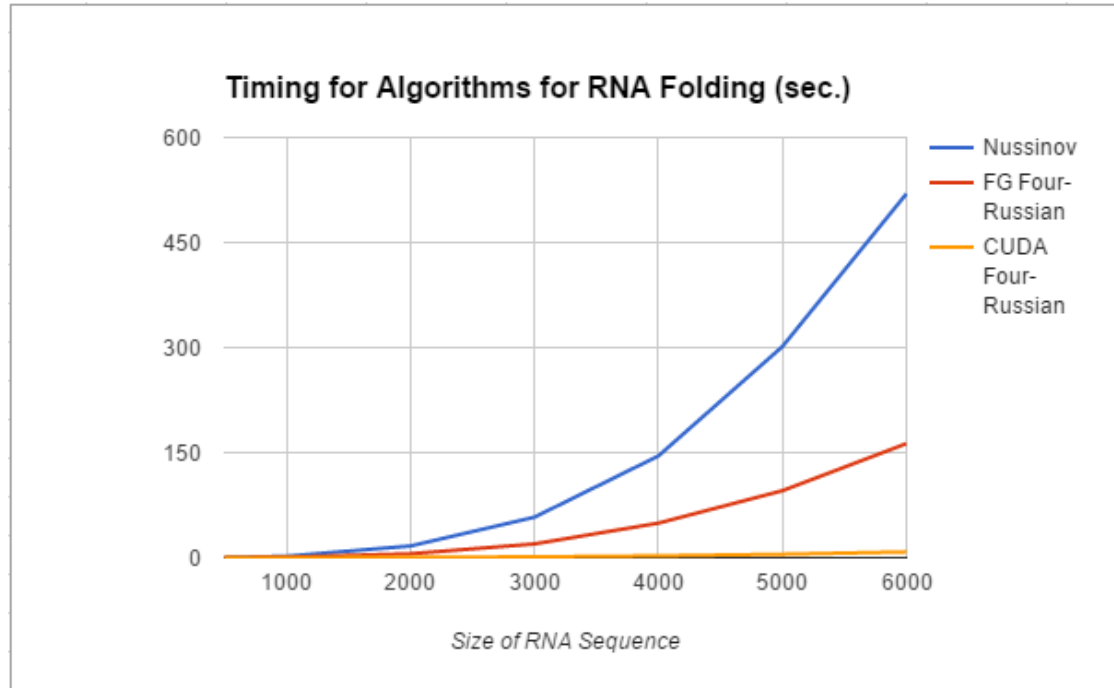


Figure 3: Timing of Algorithms for RNA Folding (sec.)

4 Discussion

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 Biology 2014:9:5.
- [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 für 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 11 1981.
- [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. j10.1000.ISBN: 978-3-642-01969-2j. jhal-00425543j
- [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.