Dynamic Programming in 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 21, 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

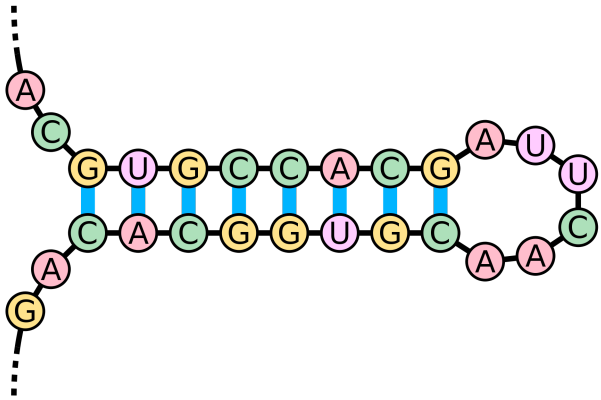

Figure 1: Example of an RNA molecule folding

up until 2010(12; 13). In 2010 the Nussinov bound that was 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 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 arbitrary length 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 also wish to determine what kind of speed difference still exists between mfold and Vienna when performing analysis on circular RNA.

We then performed micro-benchmarks(10) of the folding algorithms utilized in mfold and Vienna for both circular and linear RNA, and then compared that to equivalent benchmarks for the Four Russians and parallel Four Russians algorithms. The first thing we found when comparing mfold and Vienna was ... . When we compared mfold and Vienna to the Four Russians algorithms we found that without a graphics card and parallel processing capabilities a machine can see speed ups of ... in mfold and ... in Vienna for linear RNA. For circular RNA speed increases change to ... in mfold and ... in Vienna. With a graphics card and parallel processing folding clusters can see increases of up to ... for mfold and ... in Vienna. Circular RNA also sees speed bumps of ... in mfold and ... in

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.

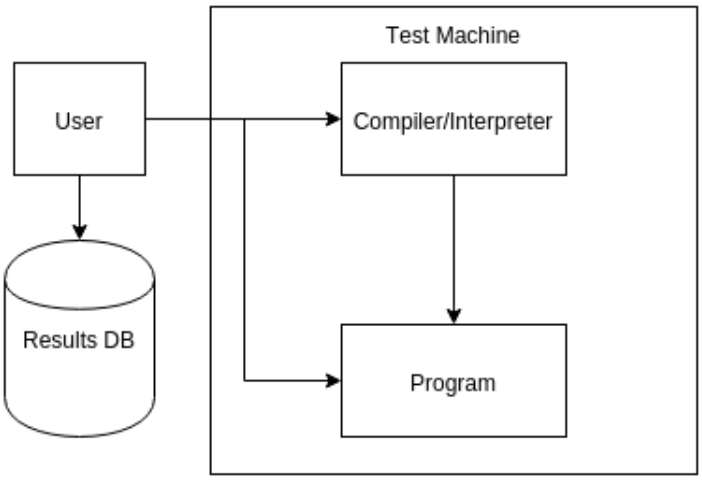

Figure 2: Test Architecture

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.

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(18). 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.

For this section we use TODO DATA

2.5 Four-russian Algorithm

Four-russian Algorithm (1) is a two-vector algorithm to improve the above-mentioned Dynamic Programming Algorithm. In 2010, it is reported that Frid and Gusfield designed an $O(\frac{n^3}{\log n})$ algorithm for RNA folding using the Four-Russians technique, which is an advance compared $O(n^3)$ Nussinov algorithm.

To apply the Four-Russians technique we start with the following observation: 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. Once the cells $D(i,l), D(i,l+1), ..., D(i,l+q-1)$ are computed, for some $l \in \{i, ..., j-q\}$, they can be represented by $D(i,l)+V_0, D(i,l)+V_1, ..., D(i,l)+V_{q-1}$, where $V_p = D(i,l+p)$, for $p \in \{0, ..., q-1\}$. Let us define, $v_0 = 0$ and $v_p = V_p - V_{p-1}$, for $p \in \{1, ..., q-1\}$. From lemma 3.1, $v_p \in \{0,1\}$, for all $p \in [0, q-1]$. Let v denote the binary vector $v_0, v_1, ..., v_{q-1}$ of differences and let V denote the vector of running totals $V_0, V_1, ..., V_{q-1}$.

Since the $v_p's$ are defined from $V_p's$, the inverse function is well defined: $V_p = \sum_{k=0}^{p} v_k$. Thus $D(i,l)$ together with the vector v represents q consecutive cells of the table. Similarly, since the values are non-increasing down a column, $D(i+l+1,j), ..., D(i+l+q,j)$ be represented by the pair $D(i+l+1,j), \bar{v}$,where $\bar{v} \in \{0,-1\}^q$.The corresponding vector of cumulative sums is denoted $\bar{v}$. We call v the horizontal difference vector or the horizontal vector and we call $\bar{v}$ the vertical difference vector or the column vector.

Thus the $D(i,j)$ can be expressed as:

$$\begin{aligned} D(i,j) &\leftarrow \max_{l+1\leq k\leq l+q} D(i,k-1)+D(k,j) \\ &\leftarrow \max_{l+1\leq k\leq l+q} D(i,l)+V_k+D(i+l+1)+\bar{V}_k \\ &\leftarrow D(i,l)++D(i+l+1,j)+\max_{l+1\leq k\leq l+q} V_k+\bar{V}_k \end{aligned}$$

By pre-processing this computation over all possible v and $\bar{v}$ vectors and tabulate the results in R, the maximum value of Table R is indexed by a pair of numbers in the range $[2q]$ to represent the two vectors $(v,\bar{v})$. The $\max_{l+1\leq k\leq l+q} V_k+\bar{V}_k$ can be determined through table lookup, which will take constant time. The operation as it fetches the max and arg max values rivals the re-calculation in Nussinov Algorithm. And this fetch will be of great advantages to the exhaustive enumeration in DP algorithm.

3 Results

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 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*. 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] 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