Proceedings of the Federated Conference on Computer Science and Information Systems pp. 449–452
DOI: 10.15439/2020F85
ISSN 2300-5963 ACSIS, Vol. 21

Efficient Computation of RNA Partition Functions Using McCaskill's Algorithm

Chunchun Zhao
Amadeus IT Group
Email: zhaochunchun@gmail.com

Sartaj Sahni
Department of Computer and Information Science and Engineering
University of Florida
Email: sahni@cise.ufl.edu

***Abstract*—We develop efficient single- and multi-core algorithms to compute partition functions for RNA sequences. Our algorithms, which are based on McCaskill's algorithm, are benchmarked against state-of-the-art fast algorithms obtained using the parallelizing source-to-source compilers PLUTO and TRACO. On our Intel I9 computational platform, our best single core algorithm takes up to 81.2% less time than the single core algorithm resulting from PLUTO, which is faster than that obtained from TRACO. Our best multi-core algorithm takes up to 84.7% less time than the multi-core algorithm obtained using TRACO when run with 20 threads (our I9 has 10 cores and supports hyperthreading); the TRACO multi-core algorithm is faster than the PLUTO one.**

I. Introduction

SEVERAL algorithms have been proposed to determine the minimum energy secondary structure of an RNA molecule. Smith and Waterman [1] and Nussinov et al. [2] have proposed a dynamic programming algorithm to maximize the number of complementary base pairs. These algorithms are oversimplified and do not generate good RNA secondary structure predictions. Zuker et al. [3] first defined five different loops that are basic units of the RNA secondary structure and proposed an energy model for each loop. Using this energy model, they developed a dynamic programming algorithm to determine the total minimum free energy structure for a given RNA sequence. The original approach of Zuker et al. [3] has been enhanced in [4], [5] by Zuker to handle more complex cases. However, Zuker's approach only provides a globally optimal secondary structure at equilibrium where each loop has been evaluated by the energy model without errors. To calculate the variance of the secondary structure, Zuker [6] proposed a further extension to calculate all suboptimal secondary structures of an RNA sequence. McCaskill [7] has proposed a totally different method to compute the full equilibrium partition function, which is the sum of the contributions of the suboptimal structures. McCaskill's algorithm computes the probabilities of each individual base pair of (i, j) in the RNA sequence. These probabilities are used, for example by the software LocaRNA [8] and PMComp/PMMulti [9] for simultaneous folding and alignment and in algorithms to predict RNA structure with a maximum expected accuracy [10], [11].

Many parallel algorithms for RNA secondary structures have also been proposed. (see, for e.g., [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]). Fekete et al. [22] first parallelized the McCaskill algorithm for a computer cluster. More recently, Palkowski and Bielecki [23] used the parallelizing source-to-source compilers PLUTO [24], [25] and TRACO [26] to automatically generate cache efficient and multi-core codes for the McCaskill algorithm.

In this paper, we begin by rewriting McCaskill's dynamic programming equations using a single matrix rather than two as used in the original equations. Then, using the row and box computation methods proposed by us in [27] and [28] to develop cache efficient algorithms for Nussinov's and Zuker's methods, respectively, for RNA folding, we develop new algorithms to compute the partition functions of McCaskill using a single array. The performance of our algorithms is compared experimentally with that of McCaskill's original algorithm and optimized versions obtained automatically by the optimizing compilers PLUTO and TRACO. Code for the original, PLUTO, and TRACO algorithms was obtained from [23]. Our experiments indicate that we are able to reduce run time by as much as 81.2% relative to the fastest previously known single core code and by as much as 84.7% relative to the fastest multi-core code using 20 threads on a 10 core Intel I9 CPU that supports hyperthreading.

The rest of the paper is organized in the following way. McCaskill's original dynamic programming equations and corresponding algorithm are presented in Section II. In Section III, we give our rewrite of McCaskill's equations and corresponding algorithms. Experimental results are presented in Section IV. Finally, Section V presents concluding remarks.

II. McCaskill's Equations and Algorithms

Let $A[1 : n] = a_1 a_2 \cdots a_n$ be an RNA sequence. McCaskill [7] develops an $O(n^4)$ and an $O(n^3)$ algorithm to compute the partition functions of A. The $O(n^3)$ algorithm uses a simplified (constant) energy function and it is this algorithm that is the focus of [23] and this paper. Let $Q(i, j)$ be the partition function of the subsequence $A[i : j]$ and let $Q^{bp}(i, j)$ be the partition function of the subsequence $A[i : j]$ when $A[i]$ and $A[j]$ form a base pair ($Q^{bp}(i, j)$ is 0 when $A[i]$ and $A[j]$ do not form a base pair). McCaskill's simplified dynamic programming equations are:

$$Q_{i,i-1} = 1,\ 1 \leq i \leq n \tag{1}$$

Algorithm 1 McCaskill Original

```
1: for i=N-1; i>=0; i−− do
2:     for j=i+1; j<N; j++ do
3:         Q[i][j] = Q[i][j-1];
4:         for k=i; k<j-l; k++ do
5:             Q^bp[k][j] = Q[k+1][j-1] * exp(−E_bp/RT) * pair(k,j);
6:             Q[i][j] += Q[i][k-1] *Q^bp[k][j];
7:         end for
8:     end for
9: end for
```

Algorithm 2 OneArray

```
1: for d=0; d<=N; d++ do // d: index of diagonal
2:     for i=0; i<=N; i++ do // i: index of row
3:         j = d+i; // j: index of column
4:         Q[i][j] = Q[i][j-1];
5:         for k=i; k<j-l; k++ do
6:             Q[i][j] += Q[i][k-1] * Q[k+1][j-1] * exp(−E_bp/RT) * pair(k,j);
7:         end for
8:     end for
9: end for
```

$$Q_{i,j} = Q_{i,j-1} + \sum_{i \le k < j-l} Q_{i,k-1} * Q^{bp}_{k,j} \tag{2}$$

$$Q^{bp}_{i,j} = Q_{i+1,j-1} * exp(-E_{bp}/RT) * pair(i,j) \tag{3}$$

where $pair(i,j)$ is 1 if $A[i]$ and $A[j]$ are complementary base pairs such as AU, GC and GU, and 0 otherwise; E_{bp} is the fixed energy, contributed by a base pair; R is a gas constant; T is the temperature; and l is the minimum loop length.

Using these equations, the partition functions Q and Q^{bp} may be computed using the algorithm $Original$ (Algorithm 1) [23], which computes the Qs and Q^{bp}s by rows from bottom to top and within a row from left to right.

Palkowski and Bielecki [23] have experimented with optimized single core and multi core versions of Algorithm $Original$ obtained using the source-to-source parallelizing compilers PLUTO [24], [25] and TRACO [26]. We use the terms $Orig$, $Pluto$, and $Traco$ to refer to their single core single thread codes for the original algorithm and its optimization using PLUTO and TRACO, respectively.

III. Our Algorithms

A. Base Algorithm For Q Using One Triangular Array

McCaskill's dynamic programming equations for Q are easily rewritten as below by eliminating Q^{bp} from Equation 2 using Equation 3. The rewritten equations are:

$$Q_{i,i-1} = 1,\ 1 \le i \le n \tag{4}$$

$$Q_{i,j} = Q_{i,j-1} + \sum_{i \le k < j-l} Q_{i,k-1} * Q_{k+1,j-1} * exp(-E_{bp}/RT) * pair(k,j) \tag{5}$$

$Q^{bp}(i,j)$, if needed, may be computed from $Q(i+1,j-1)$ using Equation 3 once all the Qs have been computed.

Using the rewritten equations, Q may be computed using a single array as shown in Algorithm 2 ($OneArray$). This algorithm computes Q by diagonals and within a diagonal from top to bottom. It's run time is $O(n^3)$ and it uses $n(n+1)/2$ space when an upper triangular two-dimensional array [29] is used for Q. Q^{bp} may be computed from Q in $O(n^2)$ time using Equation 3.

Notice that Qs that are on the same diagonal may be computed simultaneously. So, algorithm $OneArray$ is easily parallelized. Let $OneArrayP$ be the parallel version of $OneArray$ obtained by dividing each diagonal into t tiles of approximately same size, where t is the number of parallel threads.

B. ByRow (ByRow Algorithm)

In the loop of lines 5 and 6 of Algorithm $OneArray$ (Algorithm 2), the memory accesses for $Q[k+1][j-1]$ are cache inefficient as these correspond to a column access while the memory accesses for $Q[i][k-1]$, which correspond to a row access, are cache efficient. As noted by us in [27], [28] cache efficiency is enhanced by computing the Qs by rows bottom to top rather than by diagonals. Within a row, the computation is done left to right. Algorithm 3, $ByRow$, does exactly this. Though the computation order is the same in Algorithm $ByRow$ as in Algorithm $Original$ (Algorithm 1), there is a significant difference between the two algorithms besides the fact that Algorithm $ByRow$ does not use Q_{bp} explicitly. Once an element $Q[i][j]$ has been calculated in $ByRow$, all elements to its right which are on the same row are updated. This does not happen in $Original$. Relative to Algorithm 2, Algorithm 3 eliminates the inefficient column access for Q.

Algorithm 3 ByRow

```
1: for i=N-1; i>=0; i−− do // i: index of row
2:     for k=i; k<N-l; k++ do
3:         Q[i][k] = Q[i][k-1];
4:         for j=k+l+1; j<N; j++ do
5:             Q[i][j] += Q[i][k-1] * Q[k+1][j-1] * exp(−E_bp/RT) * pair(k,j);
6:         end for
7:     end for
8: end for
```

The innermost loop of Algorithm $ByRow$ (i.e., the j loop) is easily parallelized. The resulting parallel algorithm is called $ByRowP$.

C. ByBox (ByBox Algorithm)

The most cache efficient method developed by us for dynamic programming computes the elements in the upper

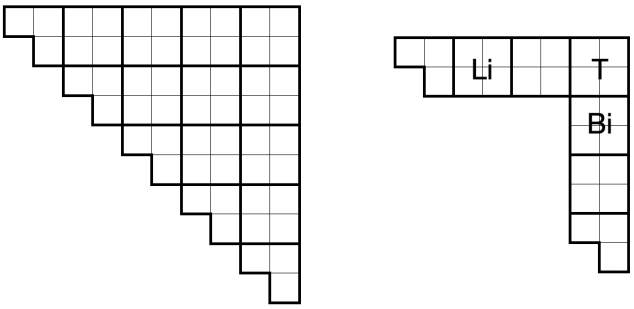

Fig. 1. Partitioning the upper triangle of Q for computation by boxes

triangle of Q by boxes rather than by rows [27], [28]. In this method, the upper triangle is divided into strips with p rows each. Each strip is divided into pxp square boxes except the leftmost partition in each strip, which is a pxp triangular box (Figure 1). p is chosen to be a multiple of the cache-line width w. Q is computed by strips bottom to top. Within a strip, the computation is done by boxes left to right. The corresponding algorithm is $ByBox$ (Algorithm 4).

Algorithm 4 ByBox

```
1: for r=N-p; r>=0; r=r-p do
2:     Calculate triangular box (r, r+p, r, r+p) using ByRow
3:     for c=r+p; c<N; c=c+p do
4:         Let T be the square box (r, r+p, c, c+p) that is to be computed
5:         Let L_0, L_1, ..., L_{k-1} be the boxes to the left of T. (L_0 is the first triangular box)
6:         Let B_1, B_2, ..., B_k be the boxes below T. (B_k is the last triangle box)
7:         for t=1; t<k; t++ do
8:             Update T using the pair (L_i, B_i)
9:         end for
10:        Update T using the pair (L_0, T) and (B_k, T).
11:    end for
12: end for
```

In our parallel version $ByBoxP$, we first compute the triangular blocks. Since these blocks are independent of one another, they may be computed in parallel using one processor per block. Next, the square blocks are computed one at a time bottom to top and within a strip left to right. When computing block T (Figure 1), multiple block pairs (L_i, B_i) can be handled simultaneously.

IV. Experimental Results

A. Experimental Platforms and Test Data

We implemented the single- and multi-core versions of the algorithms $OneArray$, $ByRow$, and $ByBox$ of Section III in C; openMP was used in the parallel codes. The performance of our codes was compared to that of the codes $Orig$, $Pluto$, and $Traco$ obtained from [23]. The tile sizes for $Pluto$ and $Traco$ used by us were the defaults (16 x 16 x 16) and (1 x 128 x 16) set by Palkowski et al. in these codes, respectively. For $ByBox$ and $ByBoxP$, the box size p was set to 32. All codes were compiled using the gcc compiler with the -O3 option and run on the platform: Intel I9-7900X Ten-Core processor 3.30GHz with 14MB LLC cache (Hyper threading supported).

For test data, we used RNA sequences obtained from the National Center for Biotechnology Information (NCBI) database [30].

B. Performance on I9

Table I gives the run times, in seconds, for the single core McCaskill algorithms on our I9 platform and shows the speedup obtained by $ByBox$ relative to each of the single core single thread algorithms for each of our test sequences. $Pluto$ is the fastest of the codes in [23] when run using a single thread and both $ByRow$ and $ByBox$ are consistently faster than $Pluto$. $ByBox$ is the fastest of our codes. The run time reduction obtained by $ByBox$ relative to $Pluto$ ranges from 41.24% to 81.22% (column labeled B vs P). $ByBox$ achieves a speedup of up to approximately 5.3 relative to $Pluto$.

Table II gives the run times, in seconds, for the multi core McCaskill algorithms on our I9 platform. Times are given using both 10 threads and 20 threads. $Traco$ was consistently faster with 20 threads than with 10 threads (recall that the I9 supports hyperthreading) and $ByBoxP$ was faster with 20 threads for sequence sizes more than 4000. $ByBoxP$ was the fastest multicore algorithm on the I9 for all sequence sizes and $Traco$ came in second. When 20 threads are used, $ByBoxP$ takes between 57.36% and 84.76% less time than does $Traco$ on our RNA sequences. Speedups of up to approximately 7.1 were obtained relative to $Traco$.

V. Conclusion

Using a rewrite of McCaskill's dynamic programming equation, we have obtained a one array implementation, $OneArray$, of McCaskill's algorithm to find the partition functions of an RNA sequence. Two cache efficient versions of $OneArray$ along with parallel multi-core versions of all three of these algorithms have been developed. On our Intel I9 computational platform, our best single core algorithm, $ByBox$, takes up to 81.2% less time than the best single core algorithm in [23] and our best multi-core algorithm, $ByBoxP$, takes up to 84.7% less time than the best multi-core algorithm in [23].

Acknowledgment

This work was supported, in part, by the National Science Foundation under award 1447711.

References

[1] M. S. Waterman and T. F. Smith, "RNA secondary structure: A complete mathematical analysis," *Mathematical Biosciences*, vol. 42, no. 3-4, pp. 257–266, 1978.

[2] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman, "Algorithms for loop matchings," *SIAM Journal on Applied mathematics*, vol. 35, no. 1, pp. 68–82, 1978.

TABLE I
RUN TIME FOR SINGLE CORE MCCASKILL ALGORITHMS, IN SECONDS, ON I9

Seq	Size	Orig	Pluto	Traco	OneArray	ByRow	ByBox	B vs P
AH001815.2	1,000	0.65	0.52	0.93	0.380	0.433	0.304	41.24%
XR_002696555.2	2,048	11.24	7.21	14.37	4.343	2.750	2.324	67.77%
NR_076111.2	3,048	44.37	29.51	50.48	24.152	10.476	7.773	73.66%
XR_002696557.1	4,026	128.87	82.39	162.11	65.872	20.427	18.281	77.81%
AJ966883.1	5,039	275.73	188.32	337.89	148.208	45.133	35.368	81.22%

TABLE II
RUN TIME FOR MULTI CORE MCCASKILL ALGORITHMS, IN SECONDS, ON I9

		10 threads					20 threads				
Seq	Size	Traco	OneArrayP	ByRowP	ByBoxP	B vs T	Traco	OneArrayP	ByRowP	ByBoxP	B vs T
AH001815.2	1,000	0.15	0.05	1.03	0.08	47.73%	0.15	0.04	1.48	0.06	57.36%
XR_002696555.2	2,048	1.72	0.42	4.87	0.35	79.39%	1.53	0.46	7.34	0.38	75.13%
NR_076111.2	3,048	6.72	2.51	11.04	1.13	83.21%	5.57	2.20	15.50	1.24	77.77%
XR_002696557.1	4,026	18.37	7.16	21.02	2.60	85.83%	15.54	6.35	30.60	2.51	83.85%
AJ966883.1	5,039	37.47	16.51	31.09	4.87	87.00%	31.27	14.02	44.98	4.77	84.76%

[3] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information," *Nucleic acids research*, vol. 9, no. 1, pp. 133–148, 1981.

[4] M. Zuker and D. Sankoff, "RNA secondary structures and their prediction," *Bulletin of mathematical biology*, vol. 46, no. 4, pp. 591–621, 1984.

[5] M. Zuker, "Computer prediction of RNA structure," in *Methods in enzymology*. Elsevier, 1989, vol. 180, pp. 262–288.

[6] ——, "On finding all suboptimal foldings of an RNA molecule," *Science*, vol. 244, no. 4900, pp. 48–52, 1989.

[7] J. S. McCaskill, "The equilibrium partition function and base pair binding probabilities for RNA secondary structure," *Biopolymers: Original Research on Biomolecules*, vol. 29, no. 6-7, pp. 1105–1119, 1990.

[8] S. Will, K. Reiche, I. L. Hofacker, P. F. Stadler, and R. Backofen, "Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering," *PLoS computational biology*, vol. 3, no. 4, 2007.

[9] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler, "Alignment of RNA base pairing probability matrices," *Bioinformatics*, vol. 20, no. 14, pp. 2222–2227, 2004.

[10] Z. J. Lu, J. W. Gloor, and D. H. Mathews, "Improved RNA secondary structure prediction by maximizing expected pair accuracy," *Rna*, vol. 15, no. 10, pp. 1805–1813, 2009.

[11] M. Palkowski and W. Bielecki, "Parallel tiled cache and energy efficient codes for o (n4) RNA folding algorithms," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 252–258, 2020.

[12] J. Li, S. Ranka, and S. Sahni, "Multicore and GPU algorithms for Nussinov RNA folding," *BMC bioinformatics*, vol. 15, no. 8, p. S1, 2014.

[13] A. Mathuriya, D. A. Bader, C. E. Heitsch, and S. C. Harvey, "GTfold: a scalable multicore code for RNA secondary structure prediction," in *Proceedings of the 2009 ACM symposium on Applied Computing*, 2009, pp. 981–988.

[14] M. S. Swenson, J. Anderson, A. Ash, P. Gaurav, Z. Sükösd, D. A. Bader, S. C. Harvey, and C. E. Heitsch, "GTfold: Enabling parallel RNA secondary structure prediction on multi-core desktops," *BMC research notes*, vol. 5, no. 1, p. 341, 2012.

[15] G. Tan, N. Sun, and G. R. Gao, "A parallel dynamic programming algorithm on a multi-core architecture," in *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, 2007, pp. 135–144.

[16] T. Estrada, A. Licon, and M. Taufer, "CompPknots: a framework for parallel prediction and comparison of RNA secondary structures with pseudoknots," in *International Symposium on Parallel and Distributed Processing and Applications*. Springer, 2006, pp. 677–686.

[17] F. Xia, Y. Dou, X. Zhou, X. Yang, J. Xu, and Y. Zhang, "Fine-grained parallel RNAalifold algorithm for RNA secondary structure prediction on FPGA," *BMC bioinformatics*, vol. 10, no. S1, p. S37, 2009.

[18] A. Jacob, J. Buhler, and R. D. Chamberlain, "Accelerating Nussinov RNA secondary structure prediction with systolic arrays on FPGAs," in *2008 International Conference on Application-Specific Systems, Architectures and Processors*. IEEE, 2008, pp. 191–196.

[19] Y. Dou, F. Xia, and J. Jiang, "Fine-grained parallel application specific computing for RNA secondary structure prediction using SCFGS on FPGA," in *Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems*, 2009, pp. 107–116.

[20] G. Rizk, D. Lavenier, and S. Rajopadhye, "GPU accelerated RNA folding algorithm," in *GPU Computing Gems Emerald Edition*. Elsevier, 2011, pp. 199–210.

[21] D.-J. Chang, C. Kimmer, and M. Ouyang, "Accelerating the Nussinov RNA folding algorithm with CUDA/GPU," in *The 10th IEEE International Symposium on Signal Processing and Information Technology*. IEEE, 2010, pp. 120–125.

[22] M. Fekete, I. L. Hofacker, and P. F. Stadler, "Prediction of RNA base pairing probabilities on massively parallel computers," *Journal of Computational Biology*, vol. 7, no. 1-2, pp. 171–182, 2000.

[23] M. Palkowski and W. Bielecki, "Parallel cache-efficient code for computing the mccaskill partition functions," in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2019, pp. 207–210.

[24] U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan, "A practical and fully automatic polyhedral program optimization system," in *ACM SIGPLAN PLDI*, vol. 10, no. 1375581.1375595, 2008.

[25] U. Bondhugula, M. Baskaran, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan, "Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model," in *International Conference on Compiler Construction*. Springer, 2008, pp. 132–146.

[26] M. Palkowski and W. Bielecki, "TRACO: source-to-source parallelizing compiler," *Computing and Informatics*, vol. 35, no. 6, pp. 1277–1306, 2017.

[27] C. Zhao and S. Sahni, "Cache and energy efficient algorithms for Nussinov's RNA folding," *BMC bioinformatics*, vol. 18, no. 15, p. 518, 2017.

[28] ——, "Efficient RNA folding using Zuker's method," in *2017 IEEE 7th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*. IEEE, 2017, pp. 1–6.

[29] S. Sahni, "Data structures, algorithms, and applications in c++, second edition." Silicon Press, 2005.

[30] "Ncbi database," http://www.ncbi.nlm.nih.gov/gquery.