LinearPartition: Linear-Time Approximation of RNA Folding Partition Function and Base Pairing Probabilities

He Zhang[a], Liang Zhang[b], David H. Mathews[c,d,e], and Liang Huang[a,b,♣]

[a]Baidu Research USA, Sunnyvale, CA 94089, USA; [b]School of Electrical Engineering & Computer Science, Oregon State University, Corvallis, OR 97330, USA; [c]Dept. of Biochemistry & Biophysics; [d]Center for RNA Biology; [e]Dept. of Biostatistics & Computational Biology, University of Rochester Medical Center, Rochester, NY 14642, USA

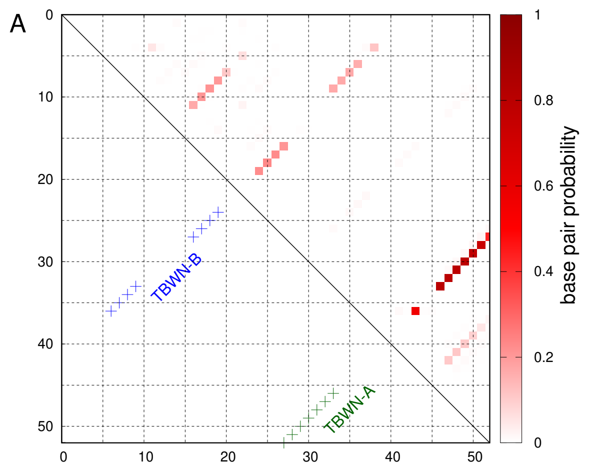
RNA secondary structure prediction is widely used to understand RNA function. Recently, there has been a shift away from the classical minimum free energy (MFE) methods to partition function-based methods that account for folding ensembles and can therefore estimate structure or base pair probabilities. However, the classic partition function algorithm scales cubically with sequence length, and is therefore a slow calculation for long sequences. This slowness is even more severe than cubic-time MFE-based methods due to a larger constant factor in runtime. Inspired by the success of LinearFold algorithm that computes the MFE structure in linear time, we address this issue by proposing a similar linear-time heuristic algorithm, LinearPartition, to approximate the partition function and base pairing probabilities. LinearPartition is 256× faster than Vienna RNAfold for a sequence with length 15,780, and 2771× faster than CONTRAfold for a sequence with length 32,753. Interestingly, although LinearPartition is approximate, it runs in linear time without sacrifice of accuracy when base pair probabilities are used to assemble structures, and even leads to a small accuracy improvement on longer families (16S and 23S rRNA).

1. Introduction

RNAs are involved in multiple processes, such as guiding RNA modifications[2] and regulating a particular disease,[3] and their functionalities are highly related to structures. However, structure determination techniques, such as X-ray crystallography[4] or Nuclear Magnetic Resonance (NMR),[5] and cryo-electron microscopy,[6] though reliable and accurate, are extremely slow and costly. Therefore, fast and accurate computational prediction of RNA structure is useful and desired. Considering full RNA structure prediction is very challenging,[7] many studies focus on predicting secondary structure, the set of canonical base pairs in the structure (A-U, G-C, G-U base pairs),[8] as it is well-defined, and provides detailed information to help understand the structure-function relationship. The secondary structure additionally is a basis to predict full tertiary structure.[9,10]

RNA secondary structure prediction is NP-complete,[11] but nested (i.e., pseudoknot-free) secondary structures can be predicted with cubic runtime dynamic programming algorithm. Commonly, the minimum free energy (MFE) structure is predicted.[12,13] At equilibrium, the MFE structure is the most populated structure, however, the MFE structure is a simplification because multiple conformations exist as an equilibrium ensemble for RNA sequences.[14] For example, many mRNAs *in vivo* form a dynamic equilibrium and fold into a population of structures.[15,16,17,18] Figure 1 shows the example of Tebowned RNA which folds into more than one structure at equilibrium. In this case, the prediction of one single structure, such as MFE structure, is not expressive enough to capture multiple states of Tebowned RNA at equilibrim.

Alternatively, we can compute the partition function, which is the sum of equilibrium constant for all possible secondary structures,



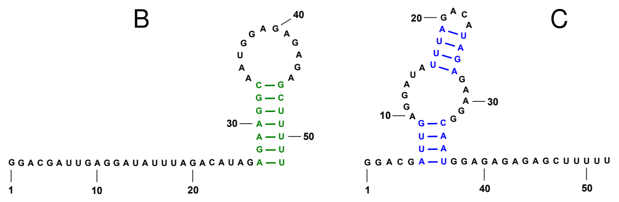


Fig. 1. An example of Tebowned RNA illustrates an RNA that folds into more than one structure at equilibrium.[1] **A**: upper triangle shows base pairing probability matrix for Tebowned RNA, and dark red squares represent high probility base pairs; lower triangle shows two different structures of Tebowned RNA at equilibrim, green crosses for TBWN-A and blue crosses for TBWN-B base pairs; **B**: TBWN-A secondary structure; **C**: TBWN-B secondary structure.

and is the normalization term for calculating the probability of a secondary structure in the Boltzmann ensemble. Starting from the partition function, we can calculate base pairing probability matrix, which stores the probabilities of each position i paired with each of possible positions j.[19,14] Figure 1A upper triangle presents the base pairing probability matrix heatmap of Tebowned RNA using Vienna RNAfold, showing that base pairs in TBWN-A have higher probabilities (in dark red) than base pairs in TBWN-B (in light red). This is consistent with the experiment result, i.e., TBWN-A is the majority structure that accounts for $56 \pm 16\%$ of the ensemble, while TBWN-B takes up $27 \pm 12\%$ of the ensemble. Partition function

Author contributions: L.H. conceived the idea and directed the project. L.H. and H.Z. designed algorithms. H.Z. wrote the Python prototype and fast C++ version. L.Z. wrote MEA code for evaluation. D.H.M. guided the evaluation that H.Z. and L.Z. carried out. H.Z. and L.H. wrote the manuscript; D.H.M. revised it.

The authors declare no conflict of interest.

♣Corresponding author: liang.huang.sh@gmail.com.

can also be applied to do stochastic sampling based on the ensemble distribution.[20,21]

In addition to model multiple states at equilibrim, base pairing probabilities are used for some downstream prediction methods, such as maximum expected accuracy (MEA),[22] to assemble a structure with improved accuracy compared with the MFE structure.[23] Some other downstream prediction methods, such as ProbKnot,[24] ThreshKnot,[25] DotKnot[26] and IPknot,[27] take base pairing probabilities to predict pseudoknotted structures with different heuristics, which is beyond the scope of standard cubic-time prediction algorithms.

Therefore, there has been a general shift from the classical MFE-based methods to partition function-based methods. These methods, as well as the prediction engines based on them, such as partition function-mode of RNAstructure,[28] Vienna RNAfold,[29] and CONTRAfold,[22] suffer the slowness from their $O(n^3)$ runtime and scale poorly for longer sequences. The slowness of partition function-based methods is even more severe than the $O(n^3)$ MFE-based methods due to its much larger runtime constant factor. For instance, for *H. pylori* 23S rRNA (sequence length 2,968 *nt*), Vienna RNAfold takes 8 seconds for the MFE structure prediction, but takes 36 seconds for partition function calculation and another 37 seconds for base pairing probabilities, which is in total 9× slower. It is even worse for CONTRAfold, which takes about 6 seconds for the MFE structure prediction, but takes 50 seconds and 70 seconds for partition function and base pairing probabilities calculation, separately, resulting to in total 20× runtime increase.

To alleviate the slowness issue, we present LinearPartition, which is inspired by the recently proposed LinearFold algorithm[30] that approximates the MFE structure in linear time. Using the same idea, LinearPartition can approximate the partition function and base pairing probability matrix in linear time. Like LinearFold, LinearPartition scans the RNA sequence from 5'-to-3' using a left-to-right fashion dynamic programming that runs in cubic time. Unlike the classical bottom-up cubic-time McCaskill algorithm,[19] our left-to-right algorithm can apply beam search pruning[31] technique to narrow down the search space, achieving linear-runtime in practice. Though the search is approximate, the well-designed heuristic makes sure the surviving structures capture the bulk part of the free energy of ensemble, and the resulting partition function is very close to the exact version.

LinearPartition is 2771× faster than CONTRAfold for the longest sequence (32,753 *nt*) ??? that CONTRAfold can run. Interestingly, LinearPartition is much faster without sacrificing accuracy when applied to downstream structure prediction tasks such as MEA and ThreshKnot (a thresholded version of ProbKnot), and even leads to a small accuracy improvement on longer families (16S and 23S rRNA).

2. Results

A. LinearPartition Algorithm. Define $\mathbf{x}$ as the input RNA sequence and $\mathcal{Y}(\mathbf{x})$ as the set of all possible secondary structures of $\mathbf{x}$. $\mathbf{y}$ is a secondary structure in $\mathcal{Y}(\mathbf{x})$. Partition function Q of $\mathbf{x}$ is defined as:

$$Q = \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} e^{-\frac{E(\mathbf{y})}{kT}} \qquad [1]$$

where $E(\mathbf{y})$ is the free energy of $\mathbf{y}$, k is the Boltzmann constant and T is the thermodynamic temperature.

$E(\mathbf{y})$ is usually calculated using loop-based "Turner" free-energy model,[32,33] but for simplicity here we adopt revised Nussinov-Jacobson energy model, i.e., a free energy of $\delta(\mathbf{x}, j)$ for unpaired base at position j and a free energy of $\xi(\mathbf{x}, i, j)$ for base pair of (i, j). For example, we can assign $\delta(\mathbf{x}, j) = 1$ kcal/mol and $\xi(\mathbf{x}, i, j) = -3$

```
1: procedure LinearPartition($\mathbf{x}$)
2:   $n \leftarrow$ length of $\mathbf{x}$
3:   $Q \leftarrow$ hash()                              ▷ hash table: from key $(i, j)$ to score
4:   $Q(0, 1) \leftarrow 1$                                                        ▷ axiom
5:   for $j = 1, ...n$ do
6:     $Q(j, j+1) \leftarrow 1$                                                    ▷ PUSH
7:     for each key $(i, j)$ in $Q$ do
8:       $Q(i, j+1) \leftarrow Q(i, j) \cdot e^{-\frac{\delta(\mathbf{x},j)}{kT}}$                          ▷ SKIP
9:       if $(x_i, x_j)$ in {AU, UA, CG, GC, GU, UG} then
10:        for each key $(k, i)$ in $Q$ do
11:          $Q(k, j+1)$ += $Q(k, i) \cdot Q(i, j) \cdot e^{-\frac{\xi(\mathbf{x},i,j)}{kT}}$         ▷ POP
12:    BeamPrune($Q, j+1$, beamsize)
13:  return $Q(0, n+1)$
```

Fig. 2. Pseudocode of a simplified version of linear-time partition function calculation algorithm. For more involved LinearPartition complete system please refer to our open source code. See Figure SI 3 for pseudocode of BEAMPRUNE procedure.

kcal/mol for CG pair, and $\xi(\mathbf{x}, i, j) = -2$ kcal/mol for AU and GU pairs. Thus, $E(\mathbf{y})$ can be decomposed as:

$$E(\mathbf{y}) = \sum_{j\in\text{unpaired}(\mathbf{y})} \delta(\mathbf{x}, j) + \sum_{(i,j)\in\text{paired}(\mathbf{y})} \xi(\mathbf{x}, i, j) \qquad [2]$$

where unpaired($\mathbf{y}$) is the set of unpaired bases in $\mathbf{y}$, and paired($\mathbf{y}$) is the set of pairs in $\mathbf{y}$.

With the simplified model, partition function Q of $\mathbf{x}$ is:

$$Q = \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \Big(\prod_{j\in\text{unpaired}(\mathbf{y})} e^{-\frac{\delta(\mathbf{x},j)}{kT}} \prod_{(i,j)\in\text{paired}(\mathbf{y})} e^{-\frac{\xi(\mathbf{x},i,j)}{kT}} \Big) \qquad [3]$$

We provide the pseudocode of our linear-time partition function algorithm (simplified version based on revised Nussinov-Jacobson energy model) in Figure 2, illustrating how our algorithm linearizes partition function calculation.

First, we define a state as:

$$\langle i, j\rangle : Q(i, j)$$

where i and j are start and end points of span $[i, j]$, and $Q(i, j)$ is the partition function of span $[i, j]$. Each state of $\langle i, j\rangle : Q(i, j)$ represents the partition function of possible substructures in span $[i, j]$. We require these substructures have at most one openning bracket at i. For example, "`(..`", "`(()`" and "`...`" are valid states, while "`((.`" and "`.(.`" are invalid.

In details of the algorithm, we use a hash Q, mapping from the key (i, j) to its partition function $Q(i, j)$, to store and look up states. $\langle 0, 1\rangle : 1.0$ represents the dummy head state, whose partition function $Q(0, 1)$ is initialized with value 1.0. LinearPartition scans from 5'-end to 3'-end (left-to-right). At each step j between 1 and n (the length of the sequence), three actions, PUSH, SKIP and POP, are called:

- PUSH: create a new state $\langle j, j+1\rangle : Q(j, j+1)$ representing an opening bracket at j, whose partition function $Q(j, j+1) = 1.0$:

$$\frac{\langle i, j\rangle : Q(i, j)}{\langle j, j+1\rangle : 1.0}$$

- SKIP: extend state $\langle i, j\rangle : Q(i, j)$ to state $\langle i, j+1\rangle : Q(i, j+1)$ by adding "`.`" on the right, where $Q(i, j+1) = Q(i, j) \cdot e^{-\frac{\delta(\mathbf{x},j)}{kT}}$:

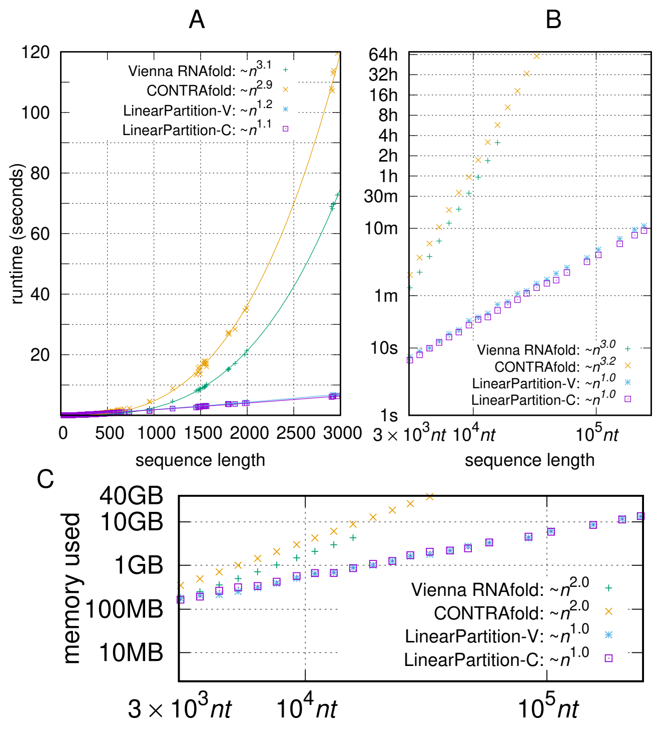


Fig. 3. Partition function and base pairing probabilities running speed and space comparisons. **A**: runtime comparisons on ArchiveII dataset; the curve-fittings were done log-log in gnuplot with $n > 10^3$. **B**: runtime comparisons on RNAcentral dataset; the x-axis and y-axis are in log scale. **C**: Memory usage comparisons on RNAcentral dataset (log scale). Note that we show the total runtime of partition function calculation and base pairing probabilities calculation. Roughly speaking, partition function calculation takes nearly the half of the total runtime.

$$\frac{\langle i, j\rangle : Q(i,j)}{\langle i, j+1\rangle : Q(i,j) \cdot e^{-\frac{\delta(\mathbf{x},j)}{kT}}}$$

- POP: when (x_i, x_j) is allowed to be paired, combine state $\langle i,j\rangle : Q(i,j)$ with its each previous state $\langle k,i\rangle : Q(k,i)$, and create a new state $\langle k, j+1\rangle : Q(k, i+1)$, where $Q(k,j+1) = Q(k,i) \cdot Q(i,j) \cdot e^{-\frac{\xi(\mathbf{x},i,j)}{kT}}$:

$$\frac{\langle k, i\rangle : Q(k,i) \qquad \langle i, j\rangle : Q(i,j)}{\langle k, j+1\rangle : Q(k,i) \cdot Q(i,j) \cdot e^{-\frac{\xi(\mathbf{x},i,j)}{kT}}}$$

  Note that the operator of updating $Q(k, j+1)$ is "+=" (see Figure 2 line 11).

Figure 2 shows that LinearPartition algorithm has three loops, one for j, one for i, and one for k, and in each loop it at most traverses n elements (i.e., k, i and j are all bounded by n). Without approximation, the search space is $O(n^3)$ as classic partition function algorithm, but in a left-to-right dynamic programming fashion instead of bottom-up. This left-to-right fashion allows to further apply beam pruning, an heuristic algorithm, and narrow down search space. The main idea is to only remain top b promissing candidates (in our case the states with higher partition function), and remove other less important ones. With beam prune, we reduce the number of states $\langle \cdot, j\rangle : Q(\cdot, j)$ to at most b, thus reduce the runtime from $O(n^3)$ to $O(nb^2)$, where b is the beam size, a user adjustable constant, and is 100 by default.

B. Efficiency and Scalability. We present two versions of LinearPartition, LinearPartition-V and LinearPartition-C. LinearPartition-V uses the experiment-based thermodynamic parameters from Vienna RNAfold[29] (https://www.tbi.univie.ac.at/RNA/download/sourcecode/2_4_x/ViennaRNA-2.4.11.tar.gz), and LinearPartition-C uses the machine learning-based parameters from CONTRAfold[22] (http://contra.stanford.edu/). Vienna RNAfold is a widely-used RNA structure prediction package, while CONTRAfold is a successful machine learning-based RNA structure prediction system. Both of them provides partition function and base pairing probabilities calculation based on classical cubic runtime algorithm. Our comparisons mainly focus on the systems with the same model, i.e., LinearPartition-V vs. Vienna RNAfold and LinearPartition-C vs. CONTRAfold. In this way the differences are based on algorithms themselves rather than models. We found a non-trival bug in CONTRAfold, which leads to overcounting in multiloop partition function calculation. We correct the bug, and all experiments are based on this bug-fixed version of CONTRAfold.

We use sequences from two datasets, ArchiveII and RNAcentral. ArchiveII dataset is first curated in the 1990s[32]and updated later with additional structures[34](http://rna.urmc.rochester.edu/pub/archiveII.tar.gz). We remove the sequences both in ArchiveII dataset and S-Processed dataset, since CONTRAfold uses S-Processed for training. We also remove 30 sequences in remaining tmRNA family because the anotation structures of these sequences contain less than 4 pseudoknots, which are suspicious. These preprocesses lead to a subset of ArchiveII with 2,859 reliable examples distributed in 9 families. Moreover, we randomly sampled 22 longer RNA sequences (no known structures) from RNAcentral (https://rnacentral.org/), with sequence lengths range from 3,048 *nt* to 244,296 *nt*. We run all experiments on a Linux machine, with 2.90GHz Intel Core i9-7920X CPU and 64G memory.

Figure 3 compares the efficiency and scalability between two baselines, Vienna RNAfold and CONTRAfold, and our two versions, LinearPartition-V and LinearPartition-C. To make the comparison fair, we disable the downstream tasks which are by default enabled with partition function and base pairing probabilities calculation, for instance, MEA structure generation in CONTRAfold and centroid structure generation in Vienna RNAfold. Figure 3A shows that both LinearPartition-V and LinearPartition-C scale almost linearly with sequence length. The runtime deviation from exact linearity is because the sequence lengths in ArchiveII dataset are relatively short. Figure 3A also confirms that the baselines scale cubically and are significantly slower than LinearPartition on long sequences. For *H. pylori* 23S rRNA sequence (2,968 *nt*, the longest sequence in ArchiveII dataset), both LinearPartition-V and LinearPartition-C take only 6 seconds, while Vienna RNAfold takes 73 seconds, and CONTRAfold is even worse, taking almost 120 seconds.

We notice that both Vienna RNAfold and CONTRAfold have limitations on sequence length. Vienna RNAfold uses a scalar estimated from minimum free energy of the given sequence to avoid overflow, but is still easy to get overflow on long sequences. For example, it overflows on the sequence with length 19,071 *nt* in the sampled RNAcentral dataset. CONTRAfold adopts logarithmic scale of partition function to solve overflow issue, but cannot run on sequence longer than 32,767 *nt* due to using "unsigned short" in its implementation. Beyond these limitations, LinearPartition can run on all sequences in RNAcentral dataset. Figure 3B visualizes the runtime of four systems on RNAcentral sampled sequences. It shows that only LinearPartition can finish all the examples, and on longer sequences the runtime of LinearPartition is exactly linear. Compar-

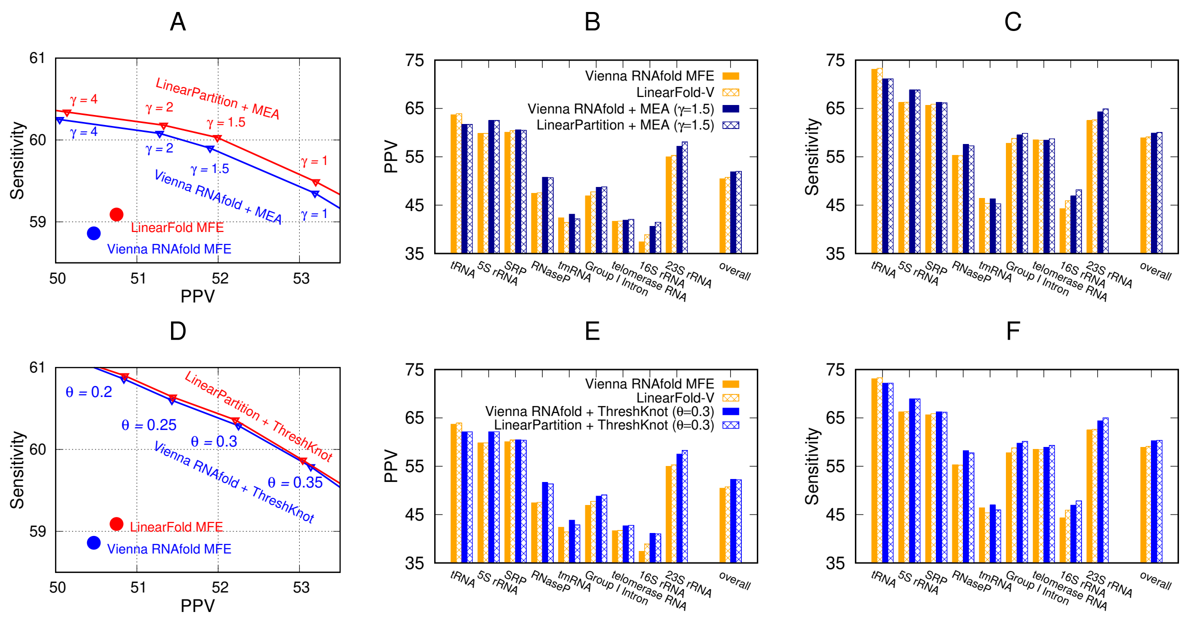


Fig. 4. Accuracy comparisons for Vienna RNAfold and LinearPartition on ArchiveII dataset. **A**: Overall MFE and MEA structure PPV-Sensitivity tradeoff of two systems with varying γ. **B** and **C**: PPV and Sensitivity comparisons of MEA structures for each family. **D**: Overall ThreshKnot structure PPV-Sensitivity tradeoff of two systems with varying threshold θ. **E** and **F**: PPV and Sensitivity comparisons of ThreshKnot structures for each family. For easy comparison, **A** and **D** are in the same scale.

ing the runtimes of the sequence with length 15,780 *nt*, the longest example showed for Vienna RNAfold in Figure 3B, Vienna RNAfold takes more than 3 hours and LinearPartition-V only takes 44.1 seconds, which is 256× faster. Note that Vienna RNAfold may not get overflow on some longer sequences, in which circumstance LinearPartition-V leads to a more salient speedup. For the longest sequence that CONTRAfold can run (32,753 *nt*) in the dataset, it takes 60.7 hours, while LinearPartition-C can finish in 52.4 seconds, which is 2771× faster, surprisingly. Even for the longest one in RNAcentral (Homo Sapiens Transcript NONHSAT168677.1 with sequence length 244,296 *nt*[35]), LinearPartition-V can finish in 10.9 minutes and LinearPartition-V can do in 9.2 minutes.

Figure 3C compares the memory usage on RNAcentral sampled sequences. It confirms that both Vienna RNAfold and CONTRAfold cost quadratic memory space, while LinearPartition costs linear space. With the length increasing, the two baselines require much more memory space than LinearPartition.

C. Accuracy.

We next consider the accuracy of using LinearPartition produced base pairing probabilities for structure prediction. First we take base pairing probability matrices from LinearPartition and Vienna RNAfold (or CONTRAfold), feed them to standard MEA algorithm separately, and compare the accuracies of prediction structures. We use Positive Predictive Value (PPV, the fraction of predicted pairs in the known structure) and sensitivity (the fraction of known pairs predicted) as accuracy measurements for each family, and get overall accuracy be averaging on families. We use slipping method to allow base pair to slip by one nucleotide.[34] We compare Vienna RNAfold and LinearPartition-V on ArchiveII dataset in the main text, and attach CONTRAfold and LinearPartition-C comparison in supporting information.

Figure 4A shows that (1) MEA-based systems (Vienna RNAfold + MEA and LinearPartition + MEA) are more accurate than MFE-based systems (Vienna RNAfold MFE and LinearFold-V); (2) LinearPartition + MEA is constantly more accurate than Vienna RNAfold + MEA. With the same γ, a hyperparameter balances PPV and sensitivity in MEA algorithm, LinearPartition + MEA enjoys a small improvement in both PPV and sensitivity.

Figures 4B and C compare PPV and sensitivity of two MFE-based systems and two MEA-based systems for each family. We can see MEA-based systems lead to accuracy improvements over MFE-based systems for most families. LinearPartition + MEA has similar PPV and sensitivity as Vienna RNAfold + MEA on short families, such as tRNA, 5S rRNA and SRP. Interestingly, LinearPartition + MEA is more accurate on long sequences, especially the two longest families, 16S and 23S Ribosomal RNAs. We also do a two-tailed permutation test, and notice that on tmRNA the MEA structures of LinearPartition is significantly worse ($p < 0.01$) than Vienna RNAfold in both PPV and Sensitivity.

ProbKnot is another partition function-based structure prediction

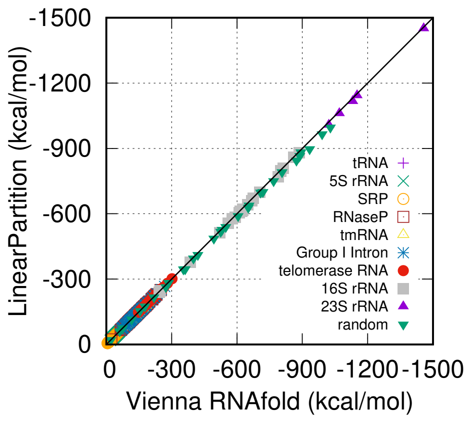


Fig. 5. Partition function approximation quality on ArchiveII dataset and some random sequences within 3,000 *nt*. x-axis and y-axis are ensemble free energy ensemble of Vienna RNAfold and LinearPartition, separately.

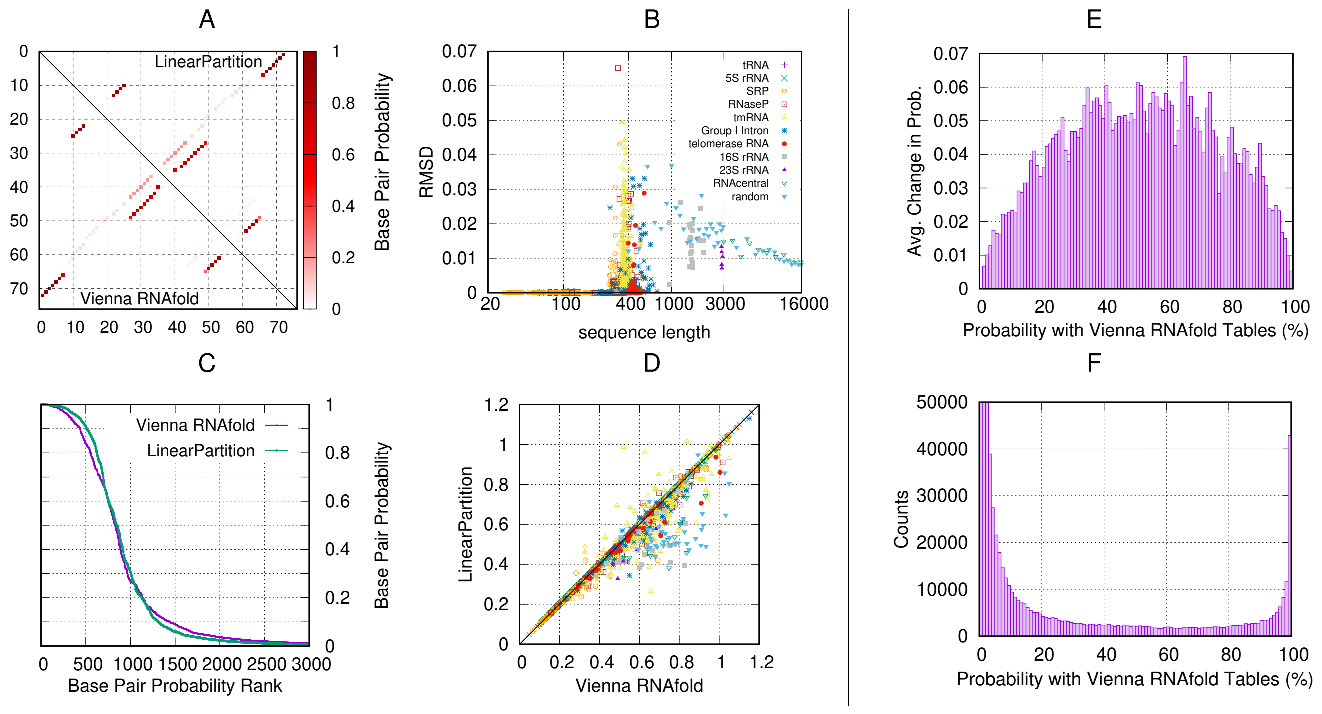


Fig. 6. Comparison of base pairing probabilities from Vienna RNAfold and LinearPartition. **A**: LinearPartition (upper triangle) and Vienna RNAfold (lower triangle) result in identical base pairing probability matrix for *E. coli* tRNA^{Gly}. **B**: root-mean-square deviation (RMSD) is relatively small between LinearPartition and Vienna RNAfold; all tRNA and 5S rRNA sequences RMSD is close to 0 (e.g., RMSD$< 10^{-5}$). **C**: LinearPartition starts higher and finishes lower than Vienna RNAfold in a sorted probability curve for *E. coli* 23S rRNA. **D**: structural entropy comparison. **E**: The mean absolute value of change in base pairing probabilities between Vienna RNAfold and LinearPartition. The changes are averaged in every base pairing probabilities bin. **F**: Plot of the pair probability distribution of Vienna RNAfold. Note that the y-axis is limited to 50,000 counts, and the counts of first three bins (with probability smaller than 3%) are far beyond 50,000.

method which adds a straightforward post-processing step after partition function calculation and is much simpler and faster than MEA. Recently, ThreshKnot, a simple thresholded version of ProbKnot, leads to more accurate predictions by filtering out unlikely pairs whose probability falls under a given threshold. It has been shown ThreshKnot can achieve better PPV and Sensitivity than the more involved MEA algorithm, and can predict pseudoknots which is beyond MEA scope, so we also compare ThreshKnot structure accuracy between Vienna RNAfold and LinearPartition.

Figure 4D shows that LinearPartition + ThreshKnot leads to a small sensitivity improvement. Figures 4E and F show that LinearPartition + ThreshKnot is sligthly better than Vienna RNAfold + ThreshKnot on long families (Group I Intron, telomerase RNA and 23S rRNA). Same as in MEA comparison, LinearPartition + ThreshKnot is significantly worse ($p < 0.01$) than Vienna RNAfold on tmRNA.

Figure SI 1 and figure SI 2 show the accuracy comparisons between CONTRAfold and LinearPartition-C with downstream tasks MEA and ThreshKnot. We can see that the results are similar as in Figure 4, i.e., LinearPartition-C is also as accurate as CONTRAfold overall, and more accurate than CONTRAfold on longer families.

Figure SI 4 employs ensemble defect to measure the average number (Figure SI 4A) and ratio (Figure SI 4B) of incorrectly predicted nucleotides.[36,37] We observe that ensemble defect of short sequences from cubic algorithm and our LinearPartition are the same or similar, but LinearPartition has lower ensemble defect for long sequences in average. This indicate that the base pairing probabilities generated by LinearPartition result in fewer predictions on incorrect base pairs.

D. Approximation Quality at Default Beam Size. Our algorithm uses beam pruning to ensure runtime and space linearity, thus is approximate compared with standard cubic algorithms. We investigate the approximation quality of LinearPartition at default beam size $b = 100$.

We first measure the approximation quality of partition function calculaion. Figure 5 shows that LinearPartition free energy of ensemble is very close to Vienna RNAfold on ArchiveII dataset and some random generated artificial RNA sequences. For short families, free energy of ensembles between LinearPartition and Vienna RNAfold are almost the same. For 23S and 16S rRNA sequences and long random sequences, LinearPartition gives a slightly lower free energy of ensemble, but the difference is very small. Random sequence's difference is a little bigger than natural sequences. Figure 5 confirms LinearPartition approximation quality of partition function is good.

Next, we measure the base pairing probabilities approximation quality using root-mean-square deviation (RMSD) between probability matrices p (from cubic algorithms, for example, Vienna RNAfold) and p' (from our algorithm, for example, LinearPartition) over the set of all possible Watson-Crick and wobble base pairs on a sequence $\mathbf{x}$. More formally, we define $\text{pairs}(\mathbf{x}) = \{1 \leq i < j \leq |\mathbf{x}| \mid \mathbf{x}_i\mathbf{x}_j \in \{\text{CG}, \text{GC}, \text{AU}, \text{UA}, \text{GU}, \text{UG}\}, \text{j} - \text{i} > 3\}$, and:

$$\text{RMSD}(p, p') = \sqrt{\frac{1}{|\text{pairs}(\mathbf{x})|} \sum_{(i,j)\in\text{pairs}(\mathbf{x})} (p_{i,j} - p'_{i,j})^2} \quad [4]$$

Figures 6A and B confirm that our LinearPartition algorithm (with default beam size 100) can indeed approximate the base pairing probability matrix reasonably well. Figure 6A shows the heatmap of

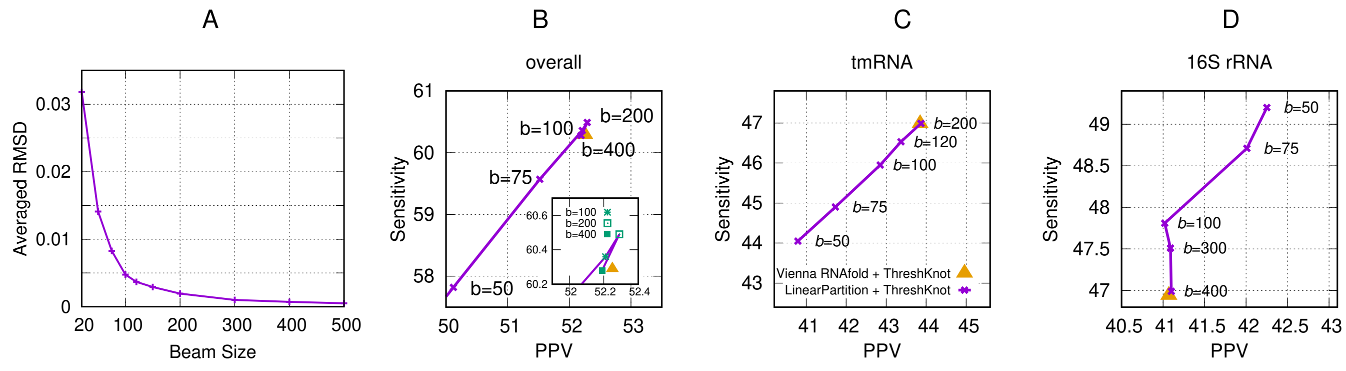


Fig. 7. Impact of beam size. **A**: RMSD changes with beam size; RMSD is averaged over all families. **B**: overall PPV and Sensitivity change with beam size. **C**: tmRNA PPV and Sensitivity change with beam size. **D**: 16S rRNA PPV and Sensitivity change with beam size. Note that the yellow triangles in **B**-**D** also denote the accuracy of LinearPartition (infinite beam size) + ThreshKnot, since LinearPartition with infinite beam size (i.e., no beam pruning) does $O(n^3)$ exact partition function calculation as Vienna RNAfold.

probability matrices for short sequence, for example, *E. coli* tRNA^{Gly} in this figure. We can see Vienna RNAfold (lower triangle) and LinearPartition (upper triangle) yield identical matrices (i.e., RMSD=0). Figure 6B shows that the RMSD of each sequence in ArchiveII and RNAcentral datasets, and random generated artificial RNA sequences, is relatively small. The highest deviation is 0.065 for *A. truei* RNaseP, which means on average each base pair's probability deviation in that worst-case sequence is about 0.065 between the cubic algorithm (Vienna RNAfold) and our linear-time one (LinearPartition). On the longest 23S rRNA family, RMSD is about 0.015. We notice that tmRNA is the family with biggest average RMSD. The random RNA sequences behave similarly to natural sequences in terms of RMSD, i.e., RMSD is very close to 0 (e.g., RMSD$< 10^{-5}$) for short ones, then becomes bigger around length 500 and decreases after that, but for most cases their RMSDs are slightly bigger compared with the natural sequences. This indicates that the approximation quality is relatively better for natural sequences. For RNAcentral sampled sequences, RMSDs are all small and around 0.01.

We assume LinearPartition base pairing probabilities distribution is peakier since it filters out states with lower free energy of ensemble in partition function calculation. We uses structural entropy[38] to measure this, where lower structural entropy indicates that the distribution is dominated by fewer base pairing probabilities. Figure 6D confirms LinearPartition distribution is peakier (lower structural entropy) than Vienna RNAfold for most sequences.

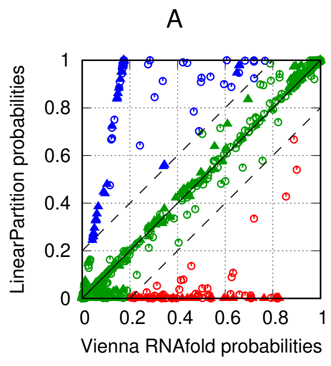
We also uses *E. coli* 23S rRNA as an example to illustrate the distribution difference. We sort all base pairing probabilities from high to low and take the top 3,000. Figure 6C shows LinearPartition distribution curve starts higher and finishes lower, confirming that its base pairing probabilities distribution is peakier.

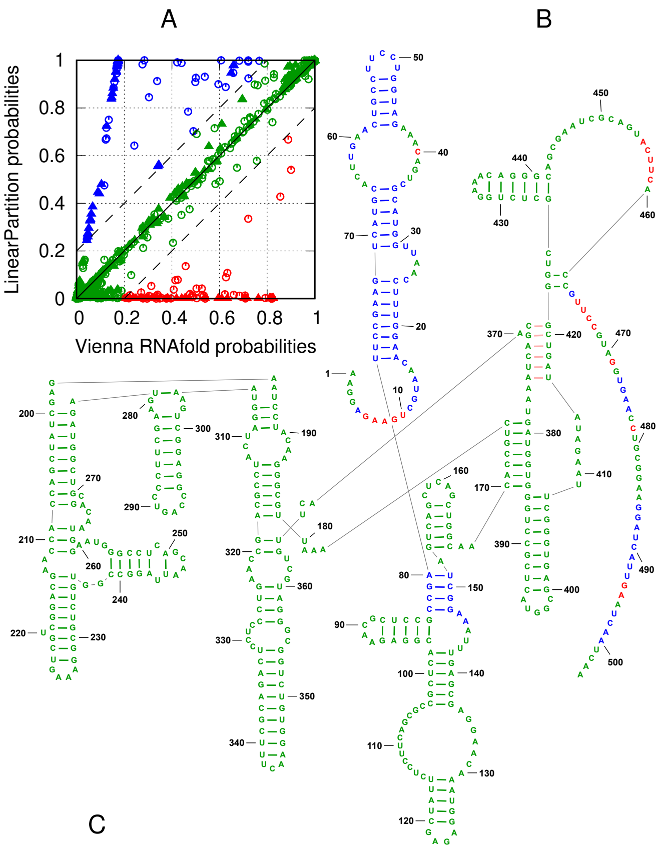
Figures 6E and F follow previous analysis method[39] to estimate approximation quality in a different perspective. We devide the base pairing probabilities range [0,1] into 100 bins, i.e., the first bin is for base pairing probabilities [0,0.01), and the second is for [0.01, 0.02), so on so forth. In Figure 6E we visualize the averaged change of base pairing probabilities between Vienna RNAfold and LinearPartition for each bin. We can see that bigger probability changes are in the middle (bins with probability around 0.5), while both on the left (bins with probability near 0) and on the right (bins with probability near 1) the changes are smaller. In Figure 6F we illustrate the counts in each bin based on Vienna RNAfold base pairing probabilities. We can see that most base pairs have very low probabilities (near 0) or very high probabilities (near 1). Combine Figures 6E and F together, we can see that probabilities of most base pairs are near 0 or 1, where the differences between Vienna RNAfold and LinearPartition are relatively small. Figure SI 5 provides the comparison of counts in each bin between Vienna RNAfold and LinearPartition-V. The count of LinearPartition-V in bin [99,100) is slightly bigger than Vienna RNAfold, while the counts in bins near 0 (being cutted at 50,000) are much less than Vienna RNAfold. This comparison also confirm that LinearPartition prunes out lots of base pairs with probabilities close to 0, and the base pairing probability distribution of LinearPartition is peakier.

E. Adjustable Beam Size. Beam size in LinearPartition is a user adjustable hyperparameter controlling beam prune, and balancing the approximation quality and runtime. Small beam size shortens runtime but sacrifices approximation quality. With the increase of beam size, LinearPartition approximates classical cubic methods and the probability matrix is finally identical to theirs when the beam size goes to infinite (no beam prune). Figure 7A confirms this analysis of beam size impact on RMSD. We observe that RMSD decreases when beam size increases. We can see even with a small beam size $b = 20$ the averaged RMSD is lower than 0.035 over all ArchiveII sequences. With default beam size $b = 100$ the averaged RMSD is lower than 0.005. With a larger beam size $b = 500$, averaged RMSD decreases to almost 0.

Beam size also has impact on PPV and Sensitivity. Figure 7B gives the overall PPV and Sensitivity changes with beam size. We can see that both PPV and Sensitivity improve from $b = 50$ to $b = 100$, and then become stable above $b = 100$. So we choose beam size 100 as the default beam size. Figures 7C and D present this impact for two selected families. Figure 7C shows that tmRNA's PPV and Sensitivity both increase when enlarging beam size. Using beam size 200, LinearPartition achieves similar PPV and Sensitivity as Vienna RNAfold. However, increasing beam size is not benefical for all families. Figure 7D gives the counterexample of 16S rRNA. We can see both PPV and Sensitivity decrease with beam size increasing from 50 to 100. After 100 Sensitivity drops with no PPV improvement.

LinearFold uses k-best parsing[40] to reduce runtime from $O(nb^2)$ to $O(nb\log b)$ without adding search error and losing accuracy. Basically, k-best parsing is to find the exact top-k (here $k = b$) states out of b^2 candidates in $O(b\log b)$ runtime by using a heap. If appling k-best parsing, LinearPartition finds and sums up the partition function of only these top-b states instead of the partition function of b^2 states. This change introduces a bigger approximation error, especially when the differences of partition function between the top-b states and the following states near the pruning boundary are small.





C

		BLUE	GREEN	RED
triangle	total	55	126645	56
	correct	23	111	0
circle	total	40	420	44
	correct	37	180	19

Fig. 8. An example of *C. ellipsoidea* Group I Intron. **A**: some high probability pairs and unpaired bases in LinearPartition have low probabilities in Vienna RNAfold (in blue), and some low probability ones in LinearPartition have high probabilities in Vienna RNAfold (in red); solid triangle stands for base pairs and unfilled circle stands for unpaired bases. **B**: ground truth structure colored with unpaired and paired probabilities from Vienna RNAfold and LinearPartition; pink binds around position 370 are pseudoknotted pairs. **C**: statistics of this example. "total" rows are the total numbers of triangles and circles with different colors in **A**, while "correct" rows are the numbers of such triangles and circles in ground truth structure.

So, in LinearPartition we do not use k-best parsing as in LinearFold, and the runtime is $O(nb^2)$ instead of $O(b\log b)$.

F. Example. We uses an RNA sequence, *C. ellipsoidea* Group I Intron (sequence length 504 *nt*) as an example to compare the base pairing probabilities generated by Vienna RNAfold and LinearPartition. In Figure 8A, we plots the unpaired bases (in circle) and base pairs (in triangle) with probabilities generated by Vienna RNAfold as x-coordinates and by LinearPartition as y-coordinates. We color the ones LinearPartition gives 0.2 higher probability than Vienna RNAfold (top left region) in blue, and color the opposite ones (bottom right region) in red. The rest ones, with probability changes smaller than 0.2 (diagonal region), are in green.

In Figure 8B, we visualize the example's ground truth structure and color the bases as in Figure 8A. We observe the majority bases are in green, indicating that Vienna RNAfold and LinearPartition agree with the main parts. But the blue helices near 5'-end indicate that LinearPartition favors these correct substructures by giving them higher probabilities than Vienna RNAfold. We also notice that all red ones (Vienna RNAfold does better than LinearPartition) are unpaired bases, which is relatively less important. This example shows that although LinearPartition gives different probabilities compared with Vienna RNAfold, it is likely that LinearPartition prediction structure is closer to ground truth structure.

Figure 8C gives the statistics of this example to further explain figure 8A and B. We can see the green triangles in figure 8A, which denote similar base pairing probabilities between Vienna RNAfold and LinearPartition, are the mojority and the total number is 126,645. The total number of blue triangles, for which LinearPartition gives higher base pairing probabilities, is 55, and among them 23 base pairs (41.82%) are in the ground truth structure. On the contrary, 56 triangles are in red, but none of these Vienna RNAfold prefered base pairs are in the ground truth structure. For unpaired bases, LinearPartition also gives more ground truth unpaired bases higher probabilities. The number of blue circles is 40, among which 37 (92.5%) are unpaired in the ground truth structure, while only 19 out of 44 red circles (43.18%) are in the ground truth structure.

3. Discussion

A. Summary. Classical partition function and base pairing probabilities calculation are "infrastructures" of many RNA studies, however, their usage is limited by slowness of cubic runtime, especially for long sequences. To address this issue, we present LinearPartition, a well-designed algorithm which can dramatically reduce runtime of partition function and base pairing probabilities calculation. We confirm that:

1. LinearPartition costs only linear runtime and memory usage, and is much faster, for example, about 2771× faster than CONTRAfold on the longest sequence (32,753 *nt*) that CONTRAfold can run in the dataset. See Figure 3.

2. Combined with downstream structure prediction methods MEA and ThreshKnot, LinearPartition leads to similar overall accuracy (or even a small improvement on MEA structures) compared with Vienna RNAfold. On long families the improvement is more pronunced. See Figure 4.

3. The approximation quality of LinearPartition is good. Although filtering out some structures, free energy of ensemble of LinearPartition is either the same or only slightly smaller than Vienna RNAfold. See Figure 5. In addition, RMSD of base pairing probabilities between LinearPartition and Vienna RNAfold is small. See Figure 6.

4. With beam size increasing, averaged RMSD decreases. The change is more pronunced from beam size 20 to 100. Above 100, averaged RMSD is smaller than 0.05, and overall PPV and Sensitivity are stable. For tmRNA, PPV and Sensitivity increase with beam size and are very close to Vienna RNAfold at beam size 200. But for 16S rRNA, accuracy drops when increase beam size. See Figure 7.

B. Extensions. Our algorithm has several potential extensions.

1. Accelerate and improve bimolecular and multistrand base pairing probabilities and accessibility. Many ncRNAs function through interacting with other RNA sequences by base pairing. Some existing methods and tools for calculating two-strands (bimolecular) or multi-strands folding partition function and base pairing

probability matrix[41,42,43,44] suffers from slowness, resulting a limitation of accessibility evaluation for long sequences. LinearPartition will provide a much faster solution for addressing this issue for these methods and tools, and will have immediate impact on their ability to predict bimolecular or multi-strand structures by improving speed and also providing additional structure information to users.

2. We will linearize the partition function-based heuristic pseudoknot prediction methods such as IPknot and Dotknot by replacing their bottleneck $O(n^3)$-time calculation of the base pairing probability matrix with our LinearPartition. All these heuristic methods use rather simple heuristic criteria to choose pairs from the base pairing probability matrix. For example, IPknot first computes base pairing probabilities and then selects base pairs using an Integer Linear Programming (ILP) methods with well-disigned constrains. Compared with solving ILP problem with efficient package such as GNU Linear Programming Kit (GLPK), computing base pairing probabilities takes more time. With LinearPartition we can overcome the costly $O(n^3)$-time calculation of the base pairing probability matrix and get an overall faster tool, FastIPknot. We can similary get FastDotKnot, etc. With these promising substantial results of LinearPartition, we believe FastIPknot (and FastDotKnot, etc) should be as accurate as, if not more accurate than, their original $O(n^3)$ versions.

ACKNOWLEDGMENTS.

References

[1] Pablo Cordero and Rhiju Das. Rich RNA structure landscapes revealed by mutate-and-map analysis. *PLOS Computational Biology*, 11(11), 2015.

[2] S. R. Eddy. Non-coding RNA genes and the modern rna world. *Nature Reviews Genetics*, 2(12):919–929, 2001.

[3] Johnny T. Y. Kung, David Colognori, and Jeannie T. Lee. Long noncoding RNAs: Past, present, and future. *Genetics*, 193(3):651–669, 2013.

[4] Sung Hou Kim, Gary Quigley, F. L. Suddath, and Alexander Rich. High-resolution x-ray diffraction patterns of crystalline transfer RNA that show helical regions. *PNAS*, 68(4):841–845, 1971.

[5] Scott L.G. and Hennig M. *RNA Structure Determination by NMR*. Humana Press, 2008.

[6] Dmitry Lyumkis. Challenges and opportunities in cryo-EM single-particle analysis. *Journal of Biological Chemistry*, 294(13):5181–5197, 2019.

[7] Zhichao Miao, Ryszard W. Adamiak, Maciej Antczak, Robert T. Batey, Alexander J. Becka, Marcin Biesiada, Michał J. Boniecki, Janusz M. Bujnicki, Shi-Jie Chen, Clarence Yu Cheng, Fang-Chieh Chou, Adrian R. Ferré-D'Amaré, Rhiju Das, Wayne K. Dawson, Feng Ding, Nikolay V. Dokholyan, Stanisław Dunin-Horkawicz, Caleb Geniesse, Kalli Kappel, Wipapat Kladwang, Andrey Krokhotin, Grzegorz E. Łach, François Major, Thomas H. Mann, Marcin Magnus, Katarzyna Pachulska-Wieczorek, Dinshaw J. Patel, Joseph A. Piccirilli, Mariusz Popenda, Katarzyna J. Purzycka, Aiming Ren, Greggory M. Rice, John Santalucia Jr., Joanna Sarzynska, Marta Szachniuk, Arpit Tandon, Jeremiah J. Trausch, Siqi Tian, Jian Wang, Kevin M. Weeks, Benfeard Williams II, Yi Xiao, Xiaojun Xu, Dong Zhang, Tomasz Zok, and Eric Westhof. RNA-puzzles round III: 3d RNA structure prediction of five riboswitches and one ribozyme. *RNA*, 23(5):655–672, 2017.

[8] I. Tinoco Jr and C. Bustamante. How RNA folds. *Journal of Molecular Biology*, 293(2):271–281, 1999.

[9] Ignacio Tinoco, Olke C. Uhlenbeck, and Mark D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230(5293):362–367, 1971.

[10] Philip E.Auron, Wayne P.Rindone, Calvin P.H. Vary, James J. Celentano, and John N. Vournakis. Computer-aided prediction of rna secondary structures. *Nucleic Acids Research*, 10:403–419., 1982.

[11] RUNE B. LYNGSØ and CHRISTIAN N.S. PEDERSEN. RNA pseudoknot prediction in energy-based models. *JOURNAL OF COMPUTATIONAL BIOLOGY*, 7(3/4):409–427, 2000.

[12] Ruth Nussinov and Ann B Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proceedings of the National Academy of Sciences*, 77(11):6309–6313, 1980.

[13] Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.

[14] David H. Mathews. Using an RNA secondary structure partition function to determine confidence in base pairs predicted by free energy minimization. *RNA*, 10(8):1178–1190, 2004.

[15] Long D, Lee R, Williams P, Chan CY, Ambros V, and Ding Y. Potent effect of target structure on microRNA function. *Nature Structural and Molecular Biology*, 14(4):287–294, 2007.

[16] Z. J. Lu and D. H. Mathews. Efficient siRNA selection using hybridization thermodynamics. *Nucleic Acids Research*, 36:640–647, 2008.

[17] Hakim Tafer, Stefan L. Ameres, Gregor Obernosterer, Christoph A. Gebeshuber, Renee Schroeder, Javier Martinez, and Ivo L. Hofacker. The impact of target site accessibility on the design of effective siRNAs. *Nature Biotechnology*, 26(5):578–583, 2008.

[18] Wan-Jung C Lai, Mohammad Kayedkhordeh, Erica V Cornell, Elie Farah, Stanislav Bellaousov, Robert Rietmeijer, David H Mathews, and Dmitri N Ermolenko. mRNAs and lncRNAs intrinsically form secondary structures with short end-to-end distances. *Nature Communications*, 9(1):4328, 2018.

[19] J. S. McCaskill. The equilibrium partition function and base pair probabilities for rna secondary structure. *Biopolymers*, 29:11105–1119, 1990.

[20] YE Ding, Chi Yu Chan, and Charles E Lawrence. RNA secondary structure prediction by centroids in a boltzmann weighted ensemble. *RNA*, 11(8):1157–1166, 2005.

[21] David H Mathews. Revolutions in RNA secondary structure prediction. *Journal of molecular biology*, 359(3):526—-532, 2006.

[22] Chuong Do, Daniel Woods, and Serafim Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, 2006.

[23] Zhi John Lu, Jason W Gloor, and David H Mathews. Improved RNA secondary structure prediction by maximizing expected pair accuracy. *RNA*, 15(10):1805–1813, 2009.

[24] Stanislav Bellaousov and David H Mathews. Probknot: fast prediction of RNA secondary structure including pseudoknots. *RNA*, 16(10):1870–1880, 2010.

[25] Liang Zhang, He Zhang, David H. Mathews, and Liang Huang. Threshknot: Thresholded probknot for improved RNA secondary structure prediction. *bioRxiv*, 2019.

[26] Jana Sperschneider and Amitava Datta. Dotknot: pseudoknot prediction using the probability dot plot under a refined energy model. *Nucleic Acids Research*, 38(7):e103–e114, 2010.

[27] Kengo Sato, Yuki Kato, Michiaki Hamada, Tatsuya Akutsu, and Kiyoshi Asai. IPknot: fast and accurate prediction of rna secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27(13):i85—-i93, 2011.

[28] David H Mathews and Douglas H Turner. Prediction of RNA secondary structure by free energy minimization. *Current Opinion in Structural Biology*, 16(3):270–278, 2006.

[29] Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):1, 2011.

[30] Liang Huang, He Zhang, Dezhong Deng, Kai Zhao, Kaibo Liu, David A Hendrix, and David H Mathews. LinearFold: linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics*, 35(14):i295–i304, 07 2019.

[31] Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*, page 1077–1086, Uppsala, Sweden, 2010. ACL.

[32] David H Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of molecular biology*, 288(5):911–940, 1999.

[33] David H. Mathews, Matthew D. Disney, Jessica L. Childs, Susan J. Schroeder, Michael Zuker, and Douglas H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences*, 101(19):7287–7292, 2004.

[34] M.F. Sloma and D.H. Mathews. Exact calculation of loop formation probability identifies folding motifs in RNA secondary structures. *RNA, 22, 1808–1818*, 2016.

[35] Yi Zhao, Hui Li, Shuangsang Fang, Yue Kang, Wei wu, Yajing Hao, Ziyang Li, Dechao Bu, Ninghui Sun, Michael Q. Zhang, and Runsheng Chen. Noncode 2016: an informative and valuable data source of long non-coding RNAs. *Nucleic Acids Research*, 44:D203–D208, 2016.

[36] Robert M. Dirks, Milo Lin, Erik Winfree, and Niles A. Pierce. Paradigms for computational nucleic acid design. *Nucleic Acids Research*, 32(4):1392–1403, 2004.

[37] JOSEPH N. ZADEH, BRIAN R. WOLFE, and NILES A. PIERCE. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, 32(3):439–452, 2010.

[38] Martijn Huynen, Robin Gutell, and Danielle Konings. Assessing the reliability of RNA folding usingstatistical mechanics. *Journal of molecular biology*, 267:1104–1112, 1997.

[39] Jeffrey Zuber, Hongying Sun, Xiaoju Zhang, Iain McFadyen, and David H. Mathews. A sensitivity analysis of RNA folding nearest neighbor parameters identifies a subset of free energy parameters with the greatest impact on RNA secondary structure prediction. *Nucleic Acids Research*, 45(10):6168–6176, 2017.

[40] Liang Huang and David Chiang. Better k-best parsing. *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 53–64, 2005.

[41] D. H. Mathews, M. E. Burkard, S. M. Freier, J. R. Wyatt, and D. H. Turner. A sequence similar to tRNA3lys gene is embedded in HIV-1 u3/r and promotes minus strand transfer. *RNA*, 5:1458—-1469, 1999.

[42] D. Piekna-Przybylska, L. DiChiacchio, D. H. Mathews, and R. A. Bambara. A sequence similar to tRNA3lys gene is embedded in HIV-1 u3/r and promotes minus strand transfer. *Nat. Struct. Mol. Biol.*, 17:83—-89, 2009.

[43] L. DiChiacchio, M. F. Sloma, and D. H. Mathews. Accessfold: predicting RNA-RNA interactions with consideration for competing self-structure. *Bioinformatics*, 32:1033—-1039, 2016.

[44] L. DiChiacchio and D. H. Mathews. *Predicting RNA-RNA interactions using RNAstructure*. Springer, 2016.

Supporting Information

LinearPartition: Linear-Time Approximation of RNA Folding Partition Function and Base Pairing Probabilities

He Zhang, Liang Zhang, David H. Mathews and Liang Huang

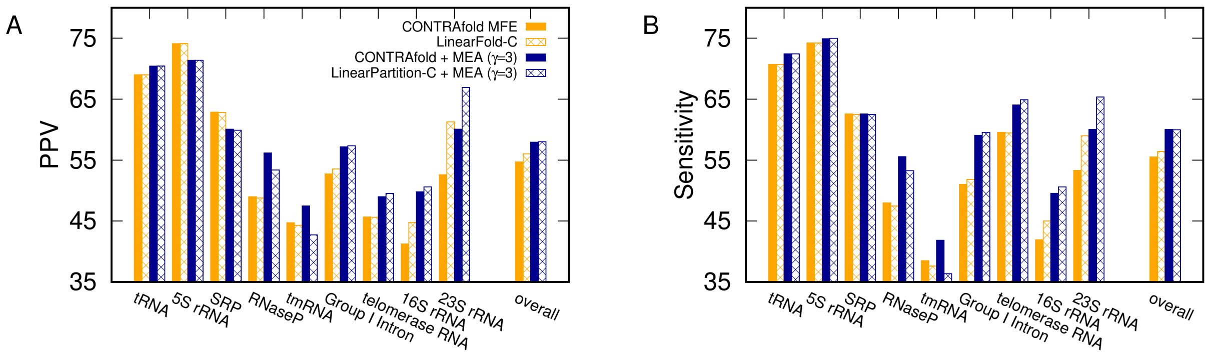


Fig. SI 1. Accuracy comparison of MEA structures ($\gamma = 3$) between CONTRAfold and LinearPartition-C on ArchiveII dataset. γ is the hyperparameter balances PPV and Sensitivity. Note that LinearPartition-C + MEA is significantly worse than CONTRAfold + MEA on two families in both PPV and Sensitivity, tmRNA and RNaseP ($p < 0.01$).

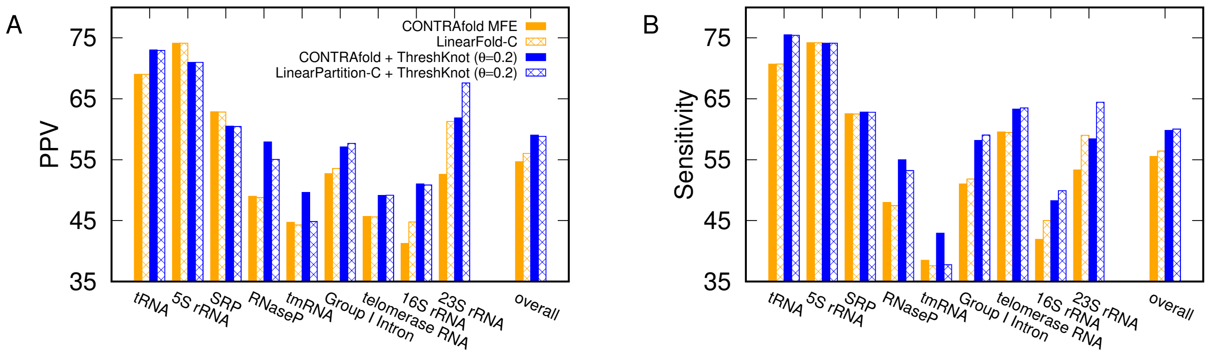


Fig. SI 2. Accuracy comparison of ThreshKnot structure ($\theta = 0.2$) between CONTRAfold and LinearPartition-C on ArchiveII dataset. θ is the hyperparameter balances PPV and Sensitivity. Note that LinearPartition-C + ThreshKnot is significantly worse than CONTRAfold + ThreshKnot on two families in both PPV and Sensitivity, tmRNA and RNaseP ($p < 0.01$), and significantly better on three longer families in Sensitivity, Group I Intron ($p < 0.01$), telomerase RNA and 16S rRNA ($0.01 \leq p < 0.05$).

```
1: procedure BEAMPRUNE($Q, j, b$)
2:     $cands \leftarrow$ hash()            ▷ hash table: from candidates $i$ to score
3:     for each key $(i, j)$ in $Q$ do
4:         $cands(i) \leftarrow Q(0, i) \cdot Q(i, j)$        ▷ $Q(0, i)$ as prefix score
5:     $cands \leftarrow$ SELECTTOPB($cands, b$)        ▷ select top-$b$ by score
6:     for each key $(i, j)$ in $Q$ do
7:         if key $i$ not in $cands$ then
8:             delete $(i, j)$ in $Q$          ▷ prune out low-scoring states
```

Fig. SI 3. Beam pruning algorithm.

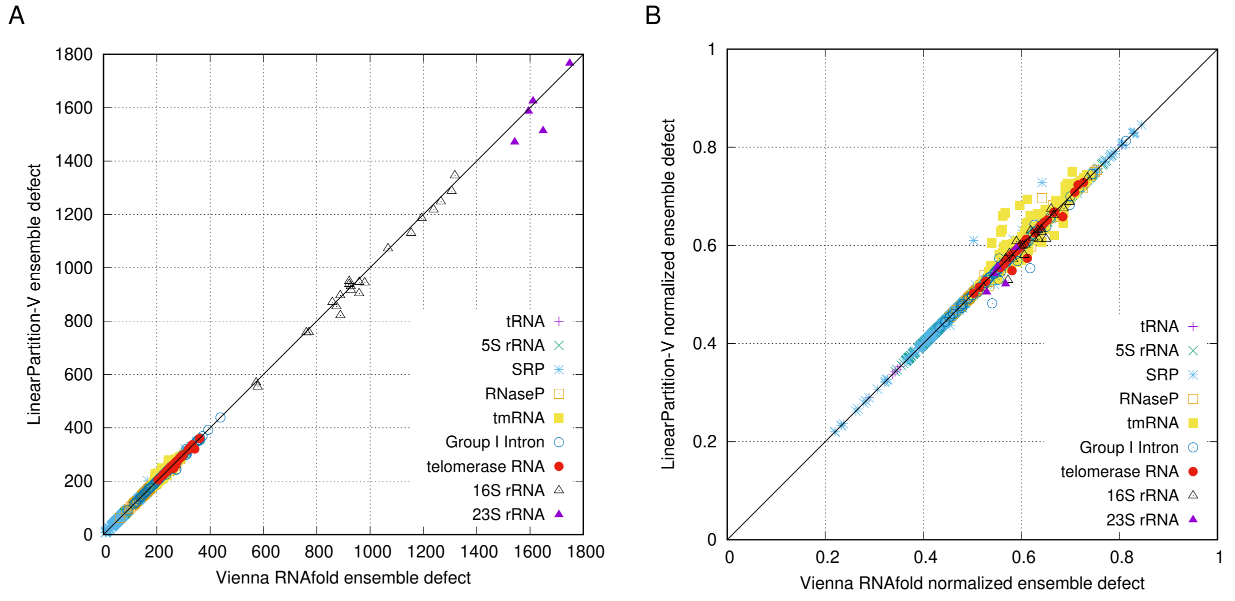


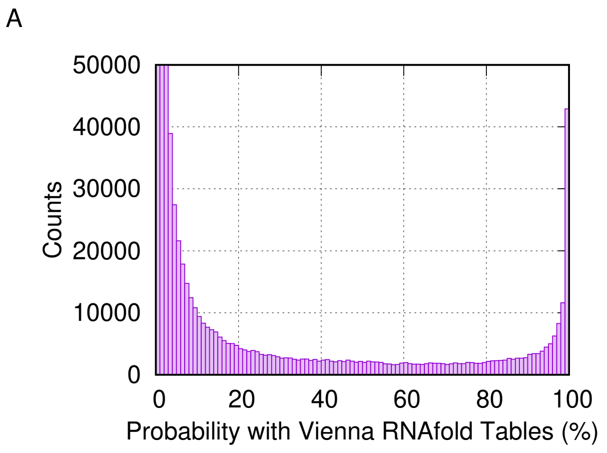
Fig. SI 4. Ensemble defect comparison of Vienna RNAfold and LinearPartition-V on ArchiveII dataset. Ensemble defect $n(\phi, s)$ indicates the averaged number of incorrectly paired nucleotides at equilibrium, and is formalized as:

$$n(\phi, s) = \sum_{\sigma \in \Gamma} p(\phi, \sigma) d(\sigma, s)$$
$$= N - \sum_{\substack{1 \leq i \leq N \\ 1 \leq j \leq N+1}} P_{i,j}(\phi) S_{i,j}(s)$$

where ϕ is the folding model, s is the groud truth secondary structure, Γ is the ensemble, σ is each possible secondary structure in Γ, and N is sequence length ($N + 1$ is for conviniently describing unpaired bases); $p(\phi, \sigma)$ is the probability of the structure σ in Γ under the folding model ϕ. $P_{i,j}(\phi)$ is the probability of i paired with j (or the probability of i being unpaired when $j = N + 1$). $S_{i,j}(s)$ is the a structure matrix with entries $S_{i,j}(s) \in \{0, 1\}$, i.e., if structure s contains pair (i, j), then $S_{i,j}(s) = 1$, otherwise $S_{i,j}(s) = 0$. $d(\sigma, s)$ is the distance between structure σ and ground truth structure s and is defined as:

$$d(\sigma, s) = N - \sum_{\substack{1 \leq i \leq N \\ 1 \leq j \leq N+1}} S_{i,j}(\sigma) S_{i,j}(s)$$

A: Ensemble defects of short sequences from two systems are equal (plots on diagnol), but ensemble defects of long sequences (16S and 23S rRNA) from LinearPartition-V are lower on average, indicating LinearPartition-V gives incorrect base pairs smaller probabilities; **B**: Ensemble defects are normalized by their sequence length. The trend is similar as **A**, but some tmRNA plots shift above diagnol.



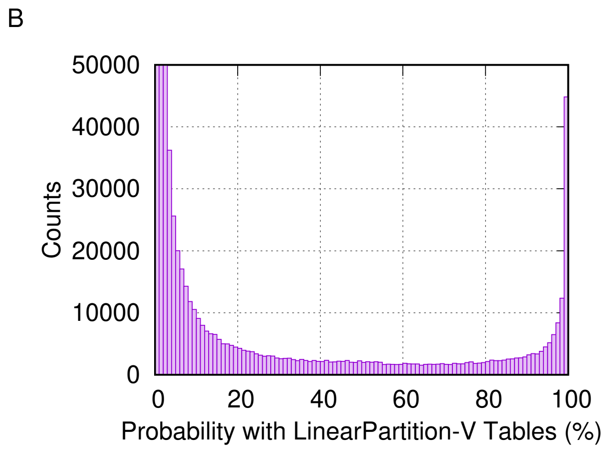


Fig. SI 5. Pair probability distributions of Vienna RNAfold and LinearPartition-V are similar. **A**: pair probability distribution of Vienna RNAfold; **B**: pair probability distribution of LinearPartition-V. The count of LinearPartition-V in bin [99,100) is slightly bigger than Vienna RNAfold, while the counts in bins near 0 (being cutted at 50,000) are much less than Vienna RNAfold.