LinearPartition: Linear-Time Approximation of RNA Folding Partition Function and Base Pair 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 48306, USA

RNA secondary structure prediction is a well-known problem, and it has been used for medical design. Compared with MFE-based methods, partition function-based methods have gained more and more attention due to their higher accuracy and ability to predict pseudoknots. However, partition function calculation, as well as the downstream base pair probability prediction, uses cubic algorithm and is slow. This slowness is even more severe than cubic MFE-based methods because of the larger cost in the inner loop. Recently, LinearFold introduced to MFE-based method with a novel dynamic programming parsing and beam search pruning borrowed from computational linguistic, and achieves even higher prediction accuracy with significantly reduced computation time. with in linear runtime and space. To further accelarate partition function-based method, we present LinearPartition, which inherits LinearFold main idea and applies it to partition function and base pair probability calculation, and leads to a small accuracy improvement in both linear runtime and linear memory space. LinearPartition reduces classical cubic runtime by pruning states with lower energy. Although it neglects some substructure, but only the ones with worse free energy are given up, and results in a similar partition function as exact search. LinearPartition is 10× faster than Vienna RNAfold for the longest sequence (about 3000 nucleotides) in the dataset. Not only fast, LinearPartition is as accurate as Vienna RNAfold when comparing MEA and ProbKnot output structure. Surprisingly, even though LinearPartition uses an inexact search, it achieves better accuracy on longer families (16S and 23S rRNA).

1. Introduction

For past decades, our understanding of ribonucleic acid (RNA) is changing. New proofs reveal that RNAs are involved in multiple processes, such as guiding RNA modifications[1] and regulating a particular disease,[2] and these functionalities are highly related to RNA's structure. Therefore, being able to determine the structure is useful and desired. However, both phisical structure determine techniques, such as X-ray crystallography[3] or Nuclear Magnetic Resonance (NMR),[4] and chemical probing methods,[5] though reliable and accurate, are extremely slow and costly, considering the exponentially increasing genomic data (about 10^{21} base-pairs per year[6]) and undetermined structures.

Due to such limitations, for many RNA tasks computational prediction of RNA structure is required. Considering whole RNA tertiary structure prediction is very challenging and even more difficult than protein folding,[7] as an alternative many studies focus on RNA secondary structure, the double helices folding structure formed by self-complementary nucleotides (A-U, G-C, G-U base pairs).[8] RNA secondary structure provides detailed information to help understand RNA's functionality,[9] which makes it an ideal foundation to further predict whole tertiary structure.[10] Furthermore, nesting secondary structure prediction problem, though still challenging, is well-defined

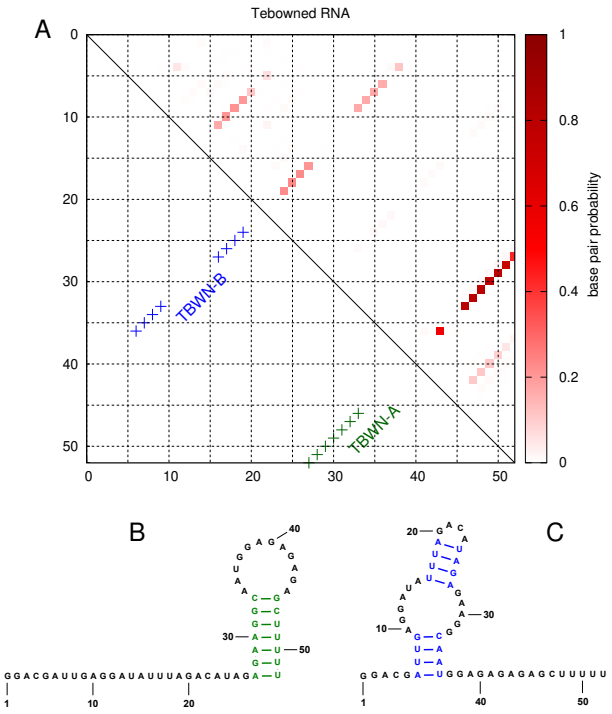

Fig. 1. An example of Tebowned RNA illustrates some RNAs fold into more than one structures at equilibrium. **A**: upper triangle shows base pair probability matrix for Tebowned RNA, and dark red squares represent high probility base pairs; lower triangle shows two different structures of Tebowned RNA at equilibrim, green cross for TBWN-A and blue cross for TBWN-B; **B**: TBWN-A drawn with StructureEditor; **C**: TBWN-B drawn with StructureEditor.

in mathematics formation, and can be suitable modeled with the decomposable substructures. Utilizing this decomposable nature, cubic runtime dynamic programming algorithm for nesting secondary structure prediction are proposed,[11] followed by an important paradigm free energy minimization (MFE) method[12] This method gives a practical solution to predict a single secondary structure, however, it neglects the facts that multiple conformations exit at equilibrium for many RNA sequences.[13] Many mRNAs *in vivo* form a dynamic equilibrium and fold into a population of structures.[14,15,16,17]

Rather than giving one single stucture prediction, partition function method takes into account possible nested structures and can model

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.

ensemble of structures at equilibrium with Boltzmann Distribution, based on which it is able to further do statistic sampling from the distribution,[18,19] and calculate base pairing probability matrix.[7,13] Base pairs with high probabilities in the matrix indicate strong confidence of pairing in prediction, and are more likely to be pairs in ground truth structures.[13,20] Figure 1 shows a real example of Tebowned RNA which folds into more than one single structures at equilibrium.[21] TBWN-A, which has a long helices at 3'-end, is the majority structure and accounts for $56 \pm 16\%$ of ensemble. While TBWN-B, which has two short helices at 5'-end, takes up $27 \pm 12\%$ of ensemble. We can see base pair probability matrix heatmap indicate base pairs in TBWN-A have higher probabilities (colored in dark red). Base pairs in TBWN-B have lower probabilities (colored in light red) than pairs in TBWN-A. Besides TBWN-A and TBWN-B illustrated in Figure 1, Tebowned RNA can also fold into the state of TBWN-C with a smaller ensemble proportion of $17 \pm 11\%$. In this case, the prediction of one single structure, such as MFE structure, is not sophisticated enough to describe multiple states of Tebowned RNA folding, while partition function and base pair probabilities estimate the folding in a different point of view and obviously model the mix of conformations fact better.

Also, base pair probabilities can be used by some downstream prediction systems, such as maximum expected accuracy (MEA),[22] to output a single nested structure with improved accuracy compared with MFE structure.[23] Some other downstream prediction systems, such as HotKnot,[24] ProbKnot,[25] DotKnot[26] and IPknot,[27] take base pair probabilities to predict pseudoknotted structure with different heuristics, and pseudoknotted structure is beyond the scope of MFE-based methods.

Therefore, there has been a general shift from the classical MFE-based methods to partition function-based methods. However, this partition function-based method, as well as prediction systems based on it such as RNAstructure,[28] CONTRAfold[22] and Vienna RNAfold,[29] suffers the slowness from its $O(n^3)$ runtime and scales poorly for longer sequences. The slowness is even more severe compared with $O(n^3)$ MFE-based methods. For instance, given *E. coli* 23S rRNA (sequence length 2900+ *nt*), Vienna RNAfold (version 2.4.11) takes about 8 seconds for MFE structure prediction, but it takes about 75 seconds for partition function and base pair probabilities calculation, which is more than 9 $\times$ slower. Recently, LinearFold[30] presents the first linear-time and linear-space MFE-based RNA folding prediction system. For the same *E. coli* 23S rRNA, LinearFold spends about 2 seconds, leading to a 4 $\times$ runtime decrease. Overall, LinearFold achieves significant efficiency and scalability improvement and higher accuracy. than classical $O(n^3)$ MFE-based method, especially on long sequences. Borrowed the efficient linearization idea from LinearFold, we presents LinearPartition, which approximates the partition function and base pair probability matrix in linear time, to address speed bottleneck in existing systems. Similar as LinearFold, LinearPartition incrementally parses a RNA sequence using a left-to-right fashion dynamic programming, and further applys beam prune[31] to narrow search space and only retain states with top b free energy of ensemble (inside score), where b is the beam size. Though introducing beam prune results to giving up some possible structures, the well-designed pruning heuristic makes sure that only structures with worse free energy of ensemble (inside score) are neglected, and partition function is still similar as exact search.

LinearPartition, inherits efficiency and accuracy of LinearFold. LinearPartition is 10$\times$ faster than the baseline Vienna RNAfold for the longest sequence (about 3000 nucleotides) in the dataset. Not only fast, LinearPartition achieves similar PPV and Sensitivity as

```
1: function LinearPartition(x)
2:     n ← length of x
3:     C ← hash()                                              ▷ chart
4:     C(0,1) ← 1                                              ▷ axiom
5:     for j = 1,...n do
6:         C(0,j+1) ← C(0,j) · e^{-δ(x,j)/kT}
7:         C(j,j+1) ← 1                                        ▷ action PUSH
8:         for all (i,j) ∈ C do
9:             C(i,j+1) ← C(i,j) · e^{-δ(x,j)/kT}              ▷ action SKIP
10:            if (x_i, x_j) ∈{AU, UA, CG, GC, GU, UG} then
11:                Q_{i,j+1} ← C(i,j) · e^{-ξ(x,i,j)/kT}
12:                for all (k,i) ∈ C do
13:                    C(k,j+1) += C(k,i) · Q_{i,j+1}          ▷ action POP
14:                end for
15:                C(0,j+1) += C(0,i) · Q_{i,j+1}              ▷ action
   COMBINE
16:            end if
17:        end for
18:        BEAMPRUNE(C, j+1,beamsize)
19:    end for
20:    return C(0,n+1)
21: end function
22:
23: function BEAMPRUNE(C, j, b)
24:    cands ← hash()                                          ▷ candidates
25:    for all (i,j) ∈ C do
26:        cands(i) ← C(0,i) · C(i,j)                          ▷ C(0,i) as prefix score
27:    end for
28:    cands ← QuickSelectTopB(cands, b)                       ▷ use quick select
   algorithm to select top-b by score
29:    for all (i,j) ∈ C do
30:        if i ∉ cands then
31:            delete (i,j) in C                               ▷ prune out low-scoring states
32:        end if
33:    end for
34: end function
```

Fig. 2. Pseudocode of a simplified version of linear-time partition function calculation algorithm, as well as a beam prune algorithm. Quick select algotithm is used in beam prune, and we skip the details for quick select here since it is well known.

Vienna RNAfold using the probability matrix computed in linear time. Surprisingly, LinearPartition leads to a significant accuracy improvement on longer families (16S and 23S rRNA).

2. Results

A. LinearPartition algorithm. We provide the pseudocode of our linear-time partition function calculation algorithm (simplified version) in Figure 2. Since complete LinearPartition system is much more involved (refer to LinearPartition open source code), here we adopt the Nussinov-like model for simplicity, focusing on how our algorithm linearizes partition function calculation in a left-to-right fashion, and omitting the details of real energy model. In the simplified model, a structure $\mathbf{y}$ of a input RNA sequence $\mathbf{x}$ is given a score of $\delta(\mathbf{x}, j)$ for unpaired nucleotide at position j, and given a score of $\xi(\mathbf{x}, i, j)$ for base pair (i, j). So the partition function Q of $\mathbf{x}$ is:

$$Q = \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \Big(\prod_{j \in \text{unpaired}(\mathbf{y})} e^{-\frac{\delta(\mathbf{x},j)}{kT}} \prod_{(i,j) \in \text{paired}(\mathbf{y})} e^{-\frac{\xi(\mathbf{x},i,j)}{kT}} \Big) \quad [1]$$

where $\mathcal{Y}(\mathbf{x})$ is the set of all possible structures of $\mathbf{x}$, k is the Boltzmann constant and T is the thermodynamic temperature.

LinearPartition scans from 5'-end to 3'-end (left-to-right), calculating the $Q_{0,j}$, which is the partial partition function from

5'-end to current step j. In details, we first define state as:

$$\langle i, j\rangle : Q_{i,j}$$

where i and j are start and end points,and $Q_{i,j}$ is the partial partition function of state $\langle i, j\rangle$. We require each state $\langle i, j\rangle$ only has at most one openning bracket of i. E.g., "`(..`" and "`...`" are valid states, while "`((.`" and "`.(`" are invalid.

We use a hash C to store and look up states. $C(0, 1)$ represents the dummy head state $\langle 0, 1\rangle$, which is initialized with value 1. At each step j from 1 to n, the length of the sequence, state $\langle 0, j + 1\rangle$ can always be extended with "`.`" from state $\langle 0, j\rangle$. Then we define four actions, PUSH, SKIP and POP, COMBINE to search states $\langle \cdot, j + 1\rangle$:

- PUSH: create a new state $\langle j, j + 1\rangle$, which has an openning bracket at j:

$$\frac{\langle i, j\rangle : Q_{i,j}}{\langle j, j + 1\rangle : 1}$$

- SKIP: extend state $\langle i, j\rangle$ to state $\langle i, j + 1\rangle$ by adding "`.`" on the right:

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

- POP: close state $\langle i, j\rangle$ when $(\mathbf{x_i}, mathbf x_j)$ is an allowed pair, and combine with all possible previous states $\langle k, i\rangle$:

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

- COMBINE: for each close state $\langle i, j\rangle$, it can be combined with its prefix state $\langle 0, i\rangle$ and form state $\langle 0, j + 1\rangle$:

$$\frac{\langle 0, i\rangle : Q_{0,i} \;\; \langle i, j\rangle : Q_{i,j}}{\langle 0, j + 1\rangle : Q_{0,i} \cdot Q_{i,j} \cdot e^{-\frac{\xi(\mathbf{x},i,j)}{kT}}}$$

Since LinearPartition algorithm has three loops in the algorithm, one for j, one for i, and one for k, and in each loop it at most traverses n elements (k, i, j are all bounded by n), it is obvious that the search space is $O(n^3)$ as classic partition function algorithm. But the left-to-right dynamic programming fashion allows to further apply beam prune on it. The main idea is to only remain top b promissing candidates, in our case the states (i, j) with higher partial partition function value $Q_{i,j}$, and remove other less promissing ones. We adopt quick select algorithm to ensure the process of selecting top b candidates costs linear runtime. With beam prune, we reduce the number of states $(\cdot, j)$ to at most b, thus reduce the runtime from $O(n^3)$ to $O(b^2n)$, where b is a user defined constant and is 100 by default.

B. Efficiency and Scalability. We present LinearPartition with two versions, LinearPartition-V and LinearPartition-C. LinearPartition-V uses the experimental-based thermodynamic parameters borrowed from Vienna RNAfold[29] (Version 2.4.11) (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 feature weights borrowed from CONTRAfold[22] (Version 2.0.2) (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 pair probabilities calculation based on classical cubic runtime algorithm.

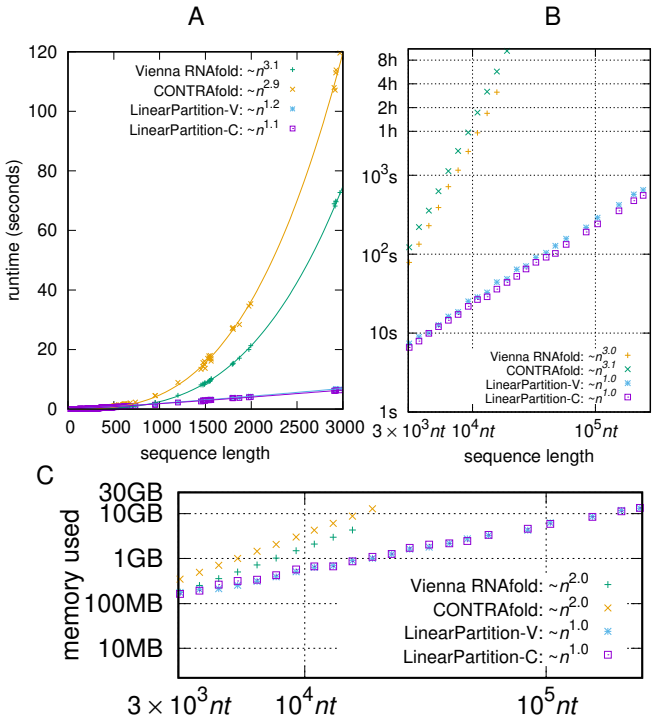

Fig. 3. Running speed and space comparison. **A**:runtime comparisons on ArchiveII dataset; the curve-fittings were done log-log in gnuplot with $n > 10^3$. **B**: runtime comparisons on samples in RNAcentral dataset; the x-axis and y-axis are in log scale. Vienna RNAfold overflows on the 18th sequence (19071 nt), so we show the first 17 sequences results for Vienna RNAfold. (CONTRAfold is still running, add later) **C**: Memory usage comparisons on samples in RNAcentral dataset (log scale).

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[33](http://rna.urmc.rochester.edu/pub/archiveII.tar.gz); We removed the sequences in ArchiveII which are also in S-Processed dataset and are used for CONTRAfold (v2.02) training. Afterthat we also removed 30 sequences in remaining tmRNA family since the anotation structures of these sequences contains less than 4 pseudoknots, which are suspicious. These preprocess leads to a subset of ArchiveII with 2859 RNA sequences and reliable structures from 9 families. We also randomly sampled 22 longer RNA sequences (no known structures) from RNAcentral (The RNAcentral Consortium, 2017) (https://rnacentral.org/), with sequence length range from 3,048 to 244,296 *nt*. We run all systems on 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 version systems, LinearPartition-V and LinearPartition-C. To make the comparison fair, we disable the downstream tasks which will be run together with partition function and base pair probability calculation by default, e.g., MEA structure generation in CONTRAfold and centroid structure generation in Vienna RNAfold. Figure 3A shows that both LinearPartition-V and LinearPartition-C scales almost linearly

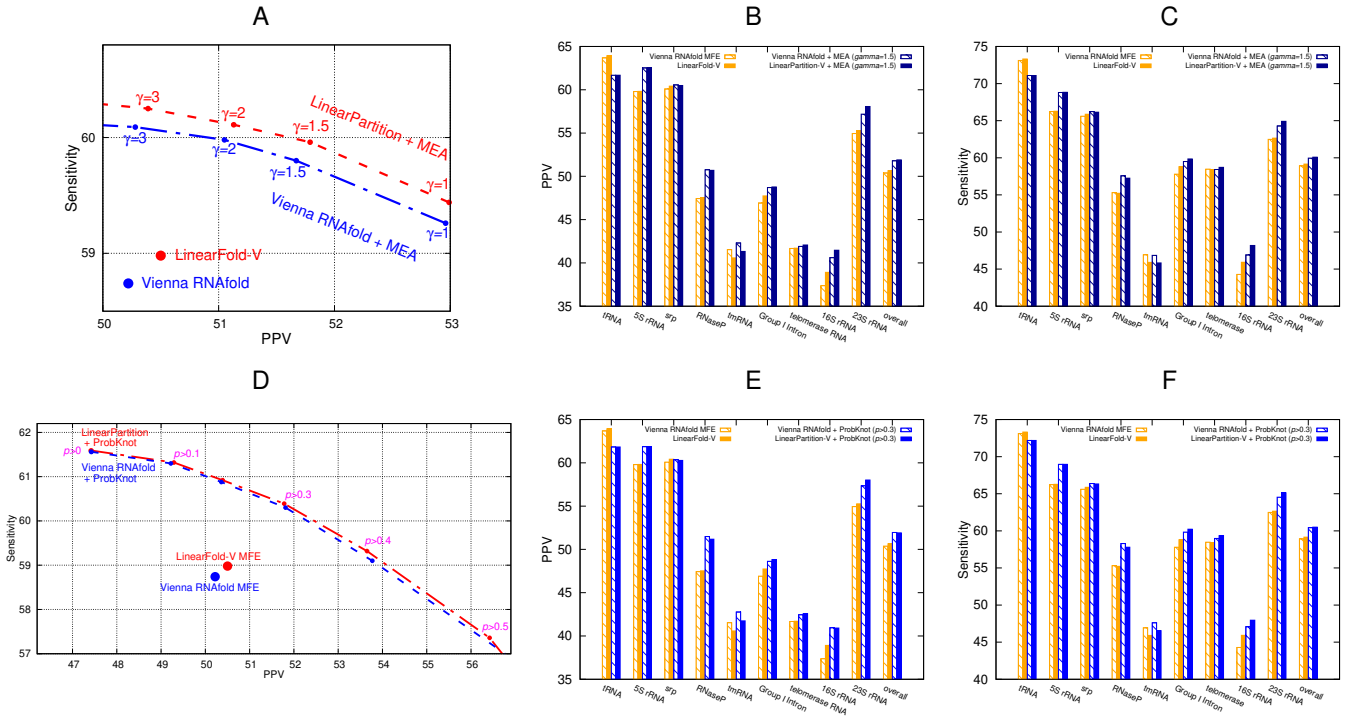

Fig. 4. Accuracy comparison for Vienna RNAfold and LinearPartition-V. **A**: Overall MFE and MEA structure PPV-sensitivity tradeoff of two systems with varying γ. **B-C**: MEA structure PPV and sensitivity comparison for each family. **D**: Overall ThreshKnot structure PPV-sensitivity tradeoff of two systems with varying threshold p. **E-F**: ThreshKnot structure PPV and sensitivity comparison for each family.

with sequence length. The runtime deviation from exact linearity is because the sequence lengths in ArchiveII dataset are relatively short (average length is 222.2 *nt*). Figure 3B confirms that on longer sequences the runtime of LinearPartition-V and LinearPartition-C are exactly linear. Figure 3A confirms that the baselines, Vienna RNAfold and CONTRAfold, scale cubically and are significantly slower than LinearPartition-V and LinearPartition-C on long sequences. For a sequence of 2,968 *nt* (23S rRNA, longest sequence in ArchiveII dataset), both LinearPartition-V and LinearPartition-C takes only 6 seconds, while Vienna RNAfold takes 75 seconds, and CONTRAfold is even worse, taking more than 110 seconds.

Figure 3B shows the runtime differences are astonishing large when running on sampled RNAcentral dataset. To avoid overflow for partition function calculation, Vienna RNAfold first calculates minimum free energy of the given sequence and estimates a scalor based on it, then applies this scalor to each inside score. However, this design not only costs some extra time on MFE calculation, but also fails to avoid overflow when applying to long sequences. For example, Vienna RNAfold overflows on the partition function calculation for the sequence with length 19,071 in our sampled RNAcentral dataset. Unlike Vienna RNAfold, LinearPartition adopts a log space partition function calculation design as CONTRAfold. Figure 3B illustrates that our LinearPartition can run on all sampled sequences in RNAcentral dataset, but Vienna RNAfold only runs on the first 17 data points because of overflow issue. Comparing the runtimes on the longest sequence (15,780 *nt*) Vienna RNAfold can run, Vienna RNAfold takes more than 3 hours and LinearPartition-V only takes 44.1 seconds, which is 256× faster. Even for the longest one (Homo Sapiens Transcript NONHSAT168677.1 with sequence length 244,296 *nt*, from the NONCODE database[34]), LinearPartition-V can predict it in 10.9 minutes. ???CONTRAfold stops at 17th or need to continue???

Figure 3C compares the memory usage in 4 systems on RNAcentral dataset. It confirms that both Vienna RNAfold and CONTRAfold cost quadratic space, while both LinearPartition-V and LinearPartition-C cost linear. With the length increase, the two baselines requires much more memory space than LinearPartition.

C. Accuracy. We next present accuracy of LinearPartition. We take the base pair probability matrices from these LinearPartition and cubic algorithm systems, and fed them to standard MEA algorithm. We use Positive Predictive Value (PPV, the fraction of predicted pairs in the known structure) and sensitivity (the fraction of known pairs predicted) to measure the accuracy across all families, as well as slipping method to allow base pair to slip by one nucleotide.[33]

Figure 4A shows that (1) MEA-based method is more accurate than MFE-based method for Vienna RNAfold and LinearPartition; (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.

Probknot is another partition-function-based method which is much simpler than MEA, only adds a linear post-processing step after the partition function calculation, and can predict pseudoknots. Recently, ThreshKnot,[?] a simple thresholded version of ProbKnot, leads to more accurate overall predictions by filtering out unlikely pairs whose prob falls under a given threshold, so we also compare ThreshKnot structure accuracy.

Figure 4D shows that LinearPartition leads to a small improvement in the downstream ThreshKnot prediction using the probability matrix computed in linear time. Figure 4E and F show that LinearPartition has similar PPV and sensitivity as the baseline, and interestingly, it is

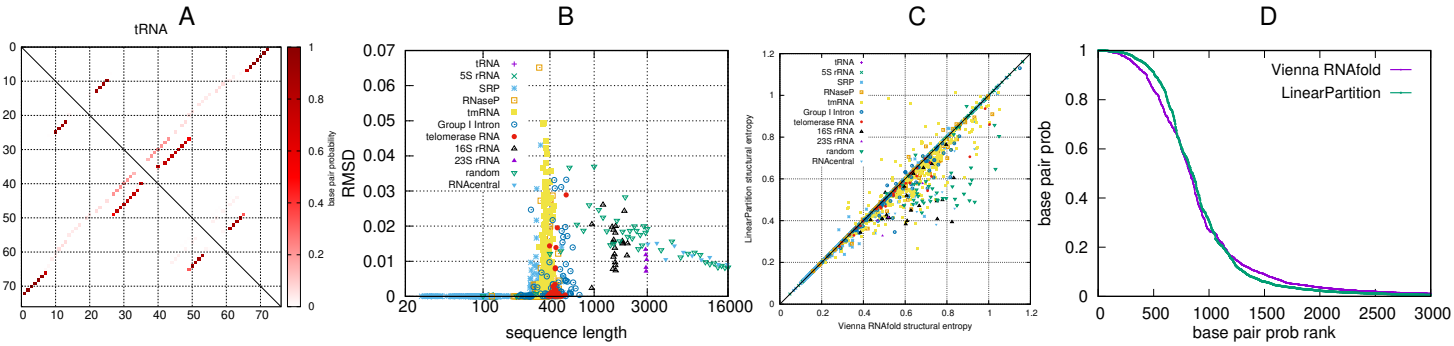

Fig. 5. Comparison of base pair probabilities from Vienna RNAfold and LinearPartition. **A**: LinearPartition (upper triangle) and Vienna RNAfold (lower triangle) result in identical base pair probability matrix for *E. coli* tRNA^{Gly}. **B**: root-mean-square deviation (RMSD) is relatively small between LinearPartition and Vienna RNAfold. **C**: structural entropy comparison. **D**: LinearPartition starts higher and finishes lower than RNAfold in a sorted probability curve for *E. coli* 23S rRNA.

more accurate on long sequences, esp. the two longest families in the database, 16S and 23S Ribosomal RNAs.

Figure SI 1 and figure SI 2 shows the accuracy comparisons between CONTRAfold and LinearPartition-C. Figure SI 1 compares MEA structures ($\gamma > 1.5$) accuracy based on these two systems and Figure SI 2 compares ThreshKnot structures ($p > 0.2$). We can see that as in Figure 4, LinearPartition-C is also more accurate than CONTRAfold on longer families.

Figure SI 3 employ ensemble defect to measure the average number (Figure SI 3A) and ratio (Figure SI 3B) of incorrectly predicted nucleotides.[35,36] 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.

D. Approximation Quality. Our algorithm uses beam prune to insure runtime and space linearity, thus is approximate compared with standard cubic algorithms. We measure the approximation quality with root-mean-square deviation (RMSD) between probability matrices p (from cubic algorithms, e.g., Vienna RNAfold) and p' (from our algorithm, e.g., LinearPartition-V) over the set of all possible pairs(x) on a sequence x (i.e., $\text{pairs}(x) = 1 \le i < j \le |x| \mid x_i x_j \in \text{CG}, \text{GC}, \text{AU}, \text{UA}, \text{GU}, \text{UG}, j - i > 3$):

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

Figure 5A and B confirm that our LinearPartition algorithm (with default beam size 100) can indeed approximate the partition function reasonably well. Figure 5A shows the heatmap of probability matrices for short sequence, e.g., *E. coli* tRNA^{Gly}, from Vienna RNAfold (lower triangle) and LinearPartition-V (upper triangle) yield identical matrices (i.e., RMSD=0). Figure 5B shows that RMSD of each sequence in ArchiveII and RNAcentral datasets, and 43 random generated artificial RNA sequences with length 100–16,000. We observe that RMSD is relatively small across all RNA families in the ArchiveII dataset. The highest deviation is 0.067 for one RNaseP sequence, which means on average, each pair's probability deviation in that worst-case sequence is about 0.067 between the exact algorithm (Vienna RNAfold) and our linear-time one (LinearPartition-V). On the longest 23S rRNA family, RMSD is about 0.015. We notice that tmRNA is the family with biggest average RMSD, and its accuracy is also the worst. The 43 random RNA sequences behave similarly to natural sequences in terms of RMSD, i.e., RMSD is 0 for short ones, and becomes bigger around length 500 and decreases then, but for most cases their RMSDs are slightly bigger compared with the natural sequences in similar length range. This indicate that the approximation quality is relatively better for natural sequences. For RNAcentral sequences, RMSDs are around 0.01.

With sequence length increasing, RMSD gradually decreases, since the number of possible pairs grows in $O(n^2)$ but the number of highly probable pairs grows in $O(n)$. To avoid RMSD is dominented by most base pairs with small probabilities, we consider a more strict circumstance. Instead of divided by the number of all possible base pairs, We only consider the ones which have probability $p > 0.01$. The RMSD resutls with such constrain are presented in Figure SI 4. It shows that for sequence shorter than $300nt$, RMSD($p > 0.01$) is still 0. This also confirms that our LinearPartition gives exactly the same probability matrix as cubic algorithm. RMSD($p > 0.01$) fluctuates from 0 to about 0.43 for sequences whose lengthes are in the range [300,1000]. Beyond 1,000nt, RMSD($p > 0.01$) becomes stable between 0.2 and 0.4.

We assume LinearPartition base pair probability distribution is peakier since it ignores low energy substructure in partition function calculation. We uses structural entropy[37] to measure this, where lower structural entropy indicates that the distribution is dominated by fewer base pairing probabilities. Figure 5C shows LinearPartition distribution is peakier (lower structural entropy) than RNAfold for most sequences.

We also uses *E. coli* 23S rRNA as an example to illustrate the distribution difference. We sort all base pair probabilities from high to low and take the top 3,000 rank. Figure 5D shows LinearPartition probability distribution curve starts higher and finishes lower, confirming that its base pair probability distribution is peakier.

E. Adjustable Beam Size. Beam size is a user adjustable hyperparameters to control beam prune, and balance the approximation quality and runtime. Small beam size shortens runtime but sacrifices approximation quality. With the increase of beam size, LinearPartition approximates optimal search and the probability matrix will be finally identical to cubic algorithms. Figure 6A shows beam size impact on RMSD. We use averaged length and averaged RMSD for each family with a certain beam size. We observe that RMSD decreases for all families with beam size increase. We can see that even with a small beam size $b = 20$ the averaged RMSD is lower than 0.04 for all families. With default beam size $b = 100$ the averaged RMSD is lower than 0.01 for all family, and tmRNA remains relatively high averaged

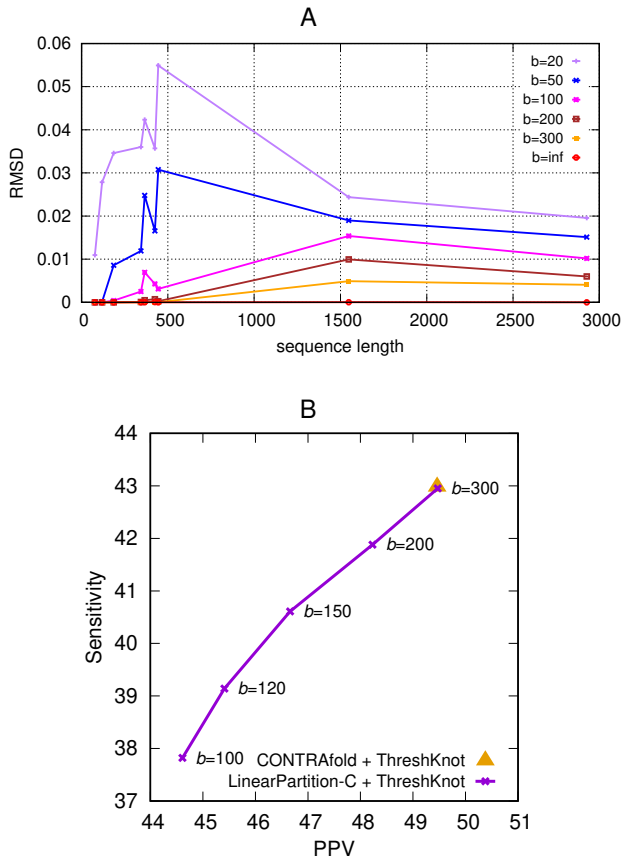

Fig. 6. Impact of beam size. **A**: RMSD change with beam size for each family; length and RMSD are averaged among all sequences in the family. **B**: tmRNA PPV and Sensitivity change with beam size.

RMSD compared with other short families. This is consistant with Figure 5B. With beam size b=infinite, which means no beam prune for partition function calculation, averaged RMSD degreases to 0 for all families. It is also clear that shorter families' averaged RMSD decreases faster.

tmRNA is the worst family in the sense of accuracy difference for LinearPartition-C. Figure 6B shows that PPV and Sensitivity both increase when enlarging beam size. When using beam size 300, LinearPartition-C achieves the same PPV and Sensitivity as CONTRAfold. This indicates that it is better to use a larger beam size, e.g., $b = 300$ when running LinearPartition-C on tmRNA sequence.

F. Example. We uses an RNA sequence as example to show secondary structure prediction accuracy of two competing system Vienna RNAfold and LinearPartition-V. The example sequence is *C.ellipsoidea* from Group I Intron family, and the sequence length is 504 *nt*. In Figure 7A, we plots all unpaired nucleotides (in circle) and base pairs (in triangle) of the example sequence's ground truth structure with probabilities calculated by Vienna RNAfold and LinearPartition-V. We colored the ones LinearPartition-V gives high probabilities but Vienna RNAfold gives low probabilities (top left region) in blue, and colored the opposite ones (bottom right region) in red. The ones that Vienna RNAfold and LinearPartition-V give similar probabilities (diagonal region) are in green. In Figure 7B, we visualize the example's ground truth structure and colored the bases as in Figure 7A. We observe the majority bases are in green, indicating that Vienna RNAfold and LinearPartition-V agree with the main part. But near 5'-end, three blue helices indicate that LinearPartition-V favors these correct substructures by giving them significant higher probabilities than Vienna RNAfold. We also notice that all Vienna RNAfold does better than LinearPartition-V are unpaired bases, which is relatively less important. This example show that although LinearPartition gives different probabilities compared with Vienna RNAfold, but it is likely that LinearPartition probabilities favors ground truth structure.

3. Discussion

A. Summary. In this paper, we present LinearPartition, which inherits LinearFold main idea and applies it to partition function and base pair probability calculation, and leads to a small accuracy improvement in both linear runtime and linear memory space. LinearPartition reduces classical cubic runtime by pruning states with lower energy. Although it neglects some substructure, but only the ones with worse free energy are given up, and results in a similar partition function as exact search. LinearPartition is 10× faster than Vienna RNAfold for the longest se- quence (about 3000 nucleotides) in the dataset. Not only fast, LinearPartition is as accurate as Vienna RNAfold when comparing MEA and ProbKnot output structure. Surprisingly, even though LinearPar- tition uses an inexact search, it achieves better accuracy on longer families (16S and 23S rRNA).

B. Analysis.

C. Extensions. Our algorithm has several potential extensions.

1. We will linearize the partition function-based heuristic pseudoknot prediction methods such as ProbKnot, IpKnot, and Dotknot by replacing their bottleneck $O(n^3)$-time calculation of the partition function with our LinearPartition. All these heuristic methods uses rather simple heuristic criteria to choose pairs from the base pair probability matrix. For example, the second step of probknot selects base pairs (i, j) where the i–j pairing probability is the largest for both bases i and j. This might appear as $O(n^2)$ in the worst case, but since the linear-time beam search used in LinearPartition only returns $O(nb)$ pairs where b is the constant beam size, this second step is still $O(n)$, giving an overall linear-time method, LinearProbKnot. We can similary get LinearIPknot, LinearProbKnot and LinearDotKnot, etc. With these promising substantial results of LinearPartition, we believe LinearProbKnot (and LinearIPknot, LinearDotKnot, etc) should be as accurate as, if not more accurate than, their original $O(n^3)$ versions.

2. Accelerate and improve bimolecular and multistrand structure prediction. Many ncRNAs function by interacting with other RNA sequences by base pairing. Some existing software tools for predicting base pairing structures between two sequences (bimolecular)[38, 39, 40, 41] suffers from slowness. LinearPartition will provide a much faster solution for the accessibility calculation, and will have immediate impact on their ability to predict bimolecular structures by improving speed and also providing additional structure information to users.

Methods

Detailed description of our algorithms, datasets, and evaluation metrics are available in the online version of the paper.

ACKNOWLEDGMENTS.

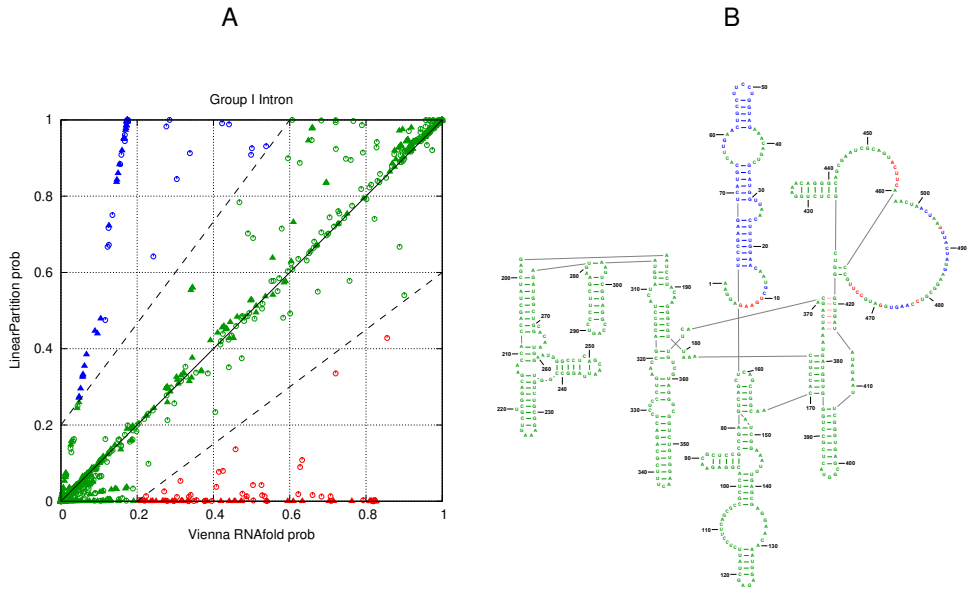

Fig. 7. An example of C. Ellipsoidea in Group I Intron family. **A**: some high probability pairs and unpaired neocleotide from ground truth structure in LinearPartition have low probabilities in RNAfold (blue plots), while some low probability ones in LinearPartition have high probabilities in RNAfold (red plots); solid triangle stands for base pair probability and unfilled circle stands for unpaired probability. **B**: ground truth structure colored with base pair and unpaired probabilities from Vienna RNAfold and LinearPartition.

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

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

[3] 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.

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

[5] William A. Ziehler and David R. Engelke. Probing RNA structure with chemical reagents and enzymes. *Current protocols in nucleic acid chemistry*, 2001.

[6] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big data: astronomical or genomical? *PLoS Biology*, 13(7):e1002195, 2015.

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

[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] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

[20] Jeffrey Zuber, B Joseph Cabral, Iain McFadyen, David M Mauger, and David H Mathews. Analysis of RNA nearest neighbor parameters reveals interdependencies and quantifies the uncertainty in RNA secondary structure prediction. *RNA*, 24(11):1568–1582, 2018.

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

[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] Jihong Ren, Baharak Rastegari, Anne Condon, and Holger H Hoos. Hotknots: heuristic prediction of RNA secondary structures including pseudoknots. *RNA*, 11(10):1494—-1504, 2005.

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

[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] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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 Pair Probabilities

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

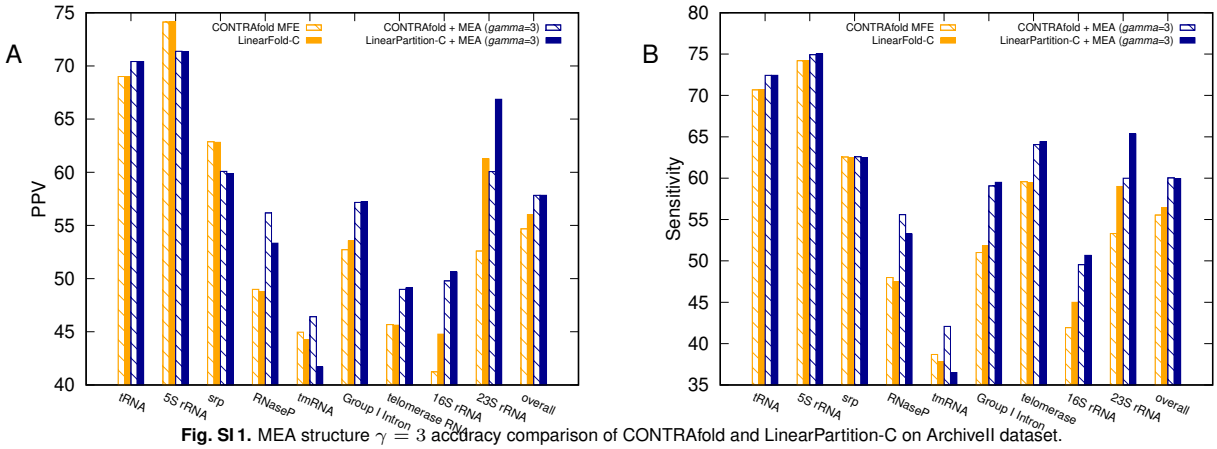

Fig. SI 1. MEA structure $\gamma = 3$ accuracy comparison of CONTRAfold and LinearPartition-C on ArchiveII dataset.

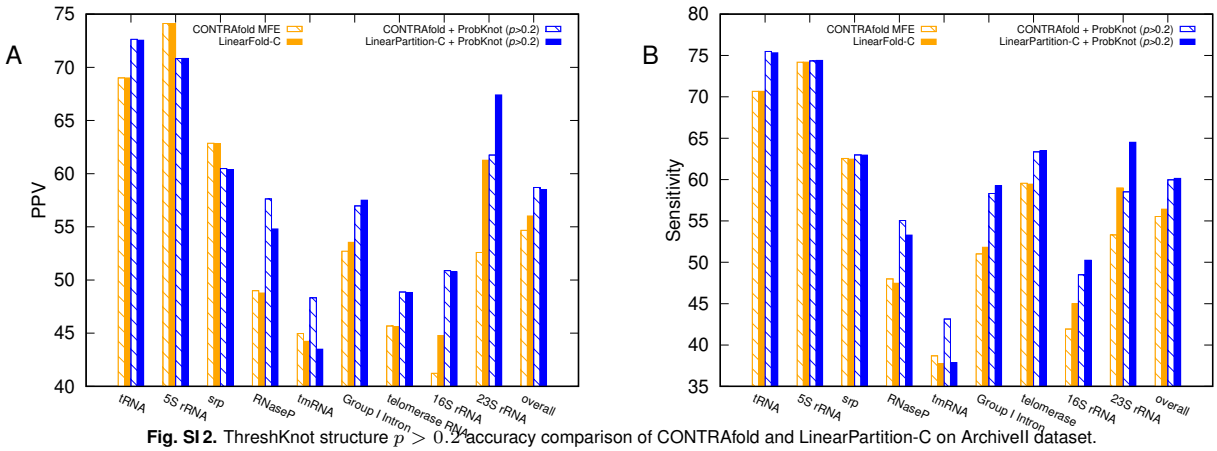

Fig. SI 2. ThreshKnot structure $p > 0.2$ accuracy comparison of CONTRAfold and LinearPartition-C on ArchiveII dataset.

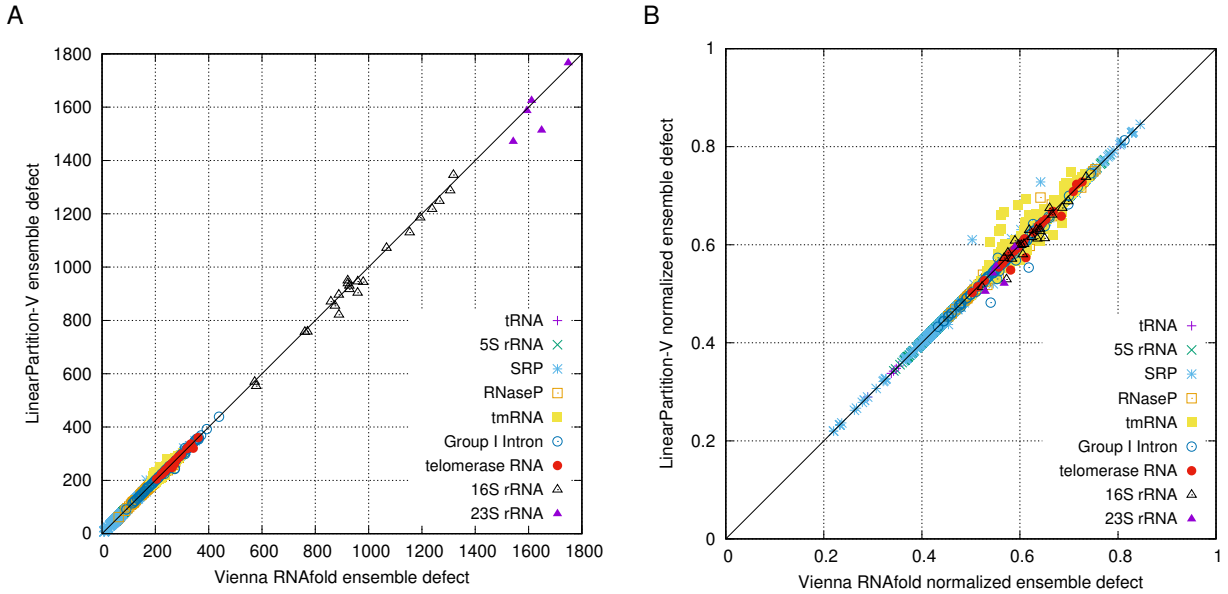

Fig. SI 3. Ensemble defect comparison of Vienna RNAfold and LinearPartition-V on ArchiveII dataset. **A**: Ensemble defects of short sequences from two systems are equal or similar (plots on diagnol), but ensemble defects of long sequences (16S and 23S rRNA) from LinearPartition-V are lower on average, 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.

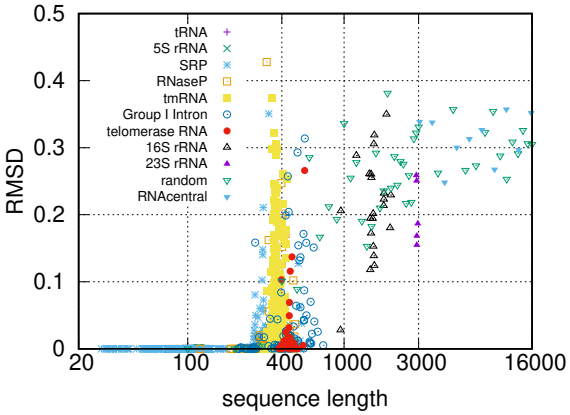

Fig. SI 4. RMSD with threshold $p > 0.01$.