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

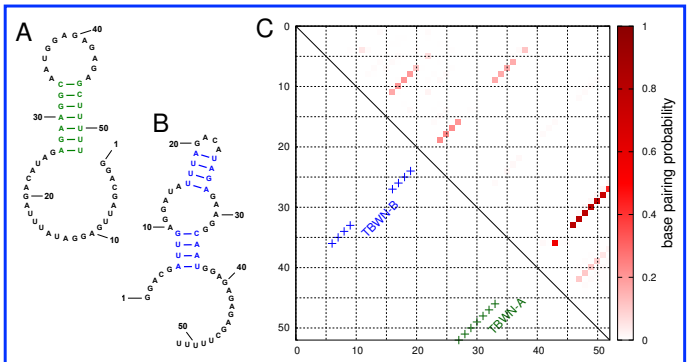
RNA secondary structure prediction is widely used to understand RNA function. Recently, there has been a shift away from the classical minimum free energy (MFE) methods to partition function-based methods that account for folding ensembles and can therefore estimate structure and base pair probabilities. 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 2,771× faster than CONTRAfold for a sequence with length 32,753. Interestingly, although LinearPartition is approximate, the resulting base pairing probabilities are better correlated with the ground truth structures and lead to a small accuracy improvement on ~~longer~~ families (16S and 23S rRNA) when used for downstream structure prediction.

1. Introduction

RNAs are involved in multiple processes, such as catalyzing reactions or guiding RNA modifications [1–3], 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 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 [18], 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 [11,19]. At equilibrium, the MFE structure is the most populated structure, but it is a simplification because multiple conformations exist as an equilibrium ensemble for RNA sequences [20]. For example, many mRNAs *in vivo* form a dynamic equilibrium and fold into a population of structures [21–24]; Figure 1A–B 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 the MFE structure, is not expressive enough to capture multiple states of RNA sequences at equilibrium.

Alternatively, we can compute the partition function, which is the sum of the equilibrium constants for all possible secondary structures, and is the normalization term for calculating the probability of a secondary structure in the Boltzmann ensemble. The partition function calculation can also be used to calculate base pairing probabilities of each nucleotide i paired with each of possible nucleotides j [12,20].



D

	span	minimum free energy		partition-function	
bottom-up, *exact*	∞	Zuker [11]	$O(n^3)$	McCaskill [12]	$O(n^3)$
local folding	L	Locafold [13]	$O(nL^2)$	RNAplfold [14] Rfold [15]	$O(nL^2)$
left-to-right, *exact*	∞	LinearFold [16]	$O(n^3)$	LinearPartition	$O(n^3)$
+ *beam pruning*			$O(nb \log b)$		$O(nb^2)$

Fig. 1. An RNA can fold into multiple structures at equilibrium. **A–B**: Two such secondary structures of Tebowned RNA: TBWN-A and TBWN-B [17]. **C**: upper triangle shows the estimated base pairing probability matrix for this RNA using Vienna RNAfold, where darker red squares represent higher probility base pairs; the lower triangle shows the two different structures; **D:** Comparison between classical, local, and left-to-right algorithms for MFE and partition function calculation. LinearFold and LinearPartition enjoy linear runtime thanks to left-to-right order ~~which~~ enables heuristic beam pruning, and both become exact $O(n^3)$ algorithms without pruning. "Span" denotes the maximum pair distance allowed (∞ means no limit); it is a small constant in local methods (e.g., default L=70 in RNAplfold).

In Figure 1C, the upper triangle presents the base pairing probability matrix of Tebowned RNA using Vienna RNAfold, showing that base pairs in TBWN-A have higher probabilities (in darker red) than base pairs in TBWN-B (in lighter red). This is consistent with the experimental result, i.e., TBWN-A is the majority structure that accounts for $56 \pm 16\%$ of the ensemble, while TBWN-B takes up $27 \pm 12\%$ [17].

In addition to model multiple states at equilibrium, base pairing probabilities are used for downstream prediction methods, such as maximum expected accuracy (MEA) [25,26], to assemble a structure with improved accuracy compared with the MFE structure [27]. Other downstream prediction methods, such as ProbKnot [28], ThreshKnot [29], DotKnot [30] and IPknot [31], use base pairing probabilities to predict pseudoknotted structures with heuristics, which is beyond the scope of standard cubic-time prediction algorithms. Additionally, the partition function is the basis of stochastic sampling, in which structures

Author contributions: L.H. conceived the idea and directed the project. H.Z., L.H., and D.H.M. designed the algorithm; H.Z. implemented it. L.Z. wrote MEA & ThreshKnot code. D.H.M. guided the evaluation that H.Z. and L.Z. carried out. H.Z., L.H., and D.H.M. wrote the manuscript.

The authors declare no conflict of interest.

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

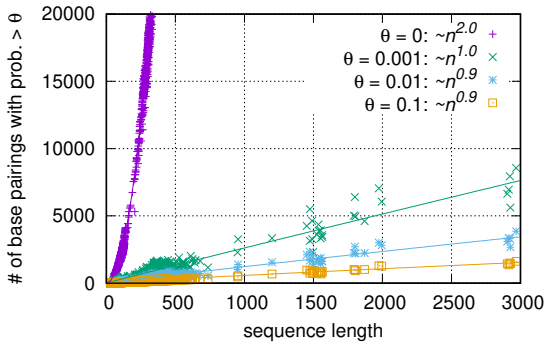


Fig. 2. Although the total number of possible base pairings scales $O(n^2)$ with the sequence length n (using the probability matrix from Vienna RNAfold as an example), with any reasonable threshold θ, the number of surviving pairings (in different colors with different θ) grows linearly, suggesting that our approximation, only computing $O(n)$ pairings, is reasonable.

```
1:  function LinearPartition(x, b)                    ▷ b is the beam size
2:    n ← length of x
3:    Q ← hash()                          ▷ hash table: from span [i, j] to Q_{i,j}
4:    Q_{j,j-1} ← 1 for all j in 1...n                              ▷ base cases
5:    for j = 1...n do
6:      for each i such that [i, j − 1] in Q do                  ▷ O(b) iterations
7:        Q_{i,j} += Q_{i,j-1} · e^{−δ(x,j)/RT}                            ▷ SKIP
8:        if x_{i-1}x_j in {AU, UA, CG, GC, GU, UG} then
9:          for each k such that [k, i − 2] in Q do                ▷ O(b) iterations
10:           Q_{k,j} += Q_{k,i-2} · Q_{i,j-1} · e^{−ξ(x,i-1,j)/RT}              ▷ POP
11:     BeamPrune(Q, j, b)                                        ▷ see Fig. SI 1
12:   return Q                                  ▷ partition function Q(x) = Q_{1,n}
```

Fig. 3. Partition function calculation pseudocode of a simplified version of the LinearPartition algorithm (the inside phase). The base-pairing probabilities are computed with the combination of the outside phase (Fig. SI 2). The actual algorithm using the Turner model is in our GitHub.

are generated at random with their probability of occurring in the Boltzmann ensemble[32,33].

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[34], Vienna RNAfold[35], and CONTRAfold[26], 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 $9\times$ 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 $20\times$ runtime increase.

To alleviate the cubic-factor slowness, we present LinearPartition, which is inspired by our recently proposed LinearFold algorithm[16] 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 dynamic program that runs in $O(n^3)$ time, but unlike the classical bottom-up McCaskill algorithm[12] with the same speed, our left-to-right scanning makes it possible to apply the beam pruning heuristic[36] to achieve linear runtime in practice; see Fig 1D. Though the search is approximate, the well-designed heuristic makes sure the surviving structures capture the bulk of the free energy of the ensemble, and the resulting partition function is close to the exact version.

More interestingly, as Figure 2 shows, even with the $O(n^3)$-time McCaskill algorithm (like the one implemented in Vienna RNAfold), the resulting number of base pairings with non-zero probabilities grows only linearly with the sequence length. This suggests that our algorithm, which only computes $O(n)$ pairings by design, is a reasonable approximation.

LinearPartition is $2,771\times$ faster than CONTRAfold for the longest sequence (32,753 *nt*) that CONTRAfold can run in the dataset. 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 (small and large subunit rRNA).

2. Results

A. LinearPartition Algorithm. We denote $\mathbf{x} = x_1 \ldots x_n$ as the input RNA sequence of length n, and $\mathcal{Y}(\mathbf{x})$ as the set of all possible secondary structures $\mathbf{y}$ of $\mathbf{x}$. The partition function $Q(\mathbf{x})$ is then:

$$Q(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} e^{-\frac{\Delta G(\mathbf{y})}{RT}} \quad [1]$$

where $\Delta G(\mathbf{y})$ is the conformational Gibbs free energy change of structure $\mathbf{y}$, R is the universal gas constant and T is the thermodynamic temperature. $\Delta G(\mathbf{y})$ is calculated using loop-based "Turner" free-energy model[37,38], but for presentation reasons, we use a revised Nussinov-Jacobson energy model, i.e., a free energy change of $\delta(\mathbf{x}, j)$ for unpaired base at position j and a free energy change 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 $\xi(\mathbf{x}, i, j) = -3$ kcal/mol for CG pair, and $\xi(\mathbf{x}, i, j) = -2$ kcal/mol for AU and GU pairs. Thus, $\Delta G(\mathbf{y})$ can be decomposed as:

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

where unpaired$(\mathbf{y})$ is the set of unpaired bases in $\mathbf{y}$, and paired$(\mathbf{y})$ is the set of base pairs in $\mathbf{y}$. The partition function now decomposes as:

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

We first define **span** $[i, j]$ to represent the subsequence $x_i \ldots x_j$ (thus $[1, n]$ denotes the whole sequence $\mathbf{x}$, and for any j in $1..n$, $[j, j-1]$ denotes the empty span between x_{j-1} and x_j). We then define a **state** to be a span associated with its partition function:

$$[i, j] : Q_{i,j}$$

where $Q_{i,j}$ encompasses all possible substructures for span $[i, j]$:

$$Q_{i,j} = \sum_{\mathbf{y} \in \mathcal{Y}(x_i \ldots x_j)} e^{-\frac{\Delta G(\mathbf{y})}{RT}}$$

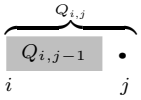
which can be visualized as:

For simplicity of presentation, in the pseudocode in Fig. 3, Q is notated as a hash table, mapping from $[i,j]$ to $Q_{i,j}$; see Supplementary Information Section A for details of its efficient implementation. As the base case, we set $Q_{j,j-1}$ to be 1 for all j, meaning all empty spans have partition function of 1 (line 4). Our algorithm then scans the sequence from left-to-right (i.e., from 5'-to-3'), and at each nucleotide x_j ($j = 1 \ldots n$), we perform two actions, SKIP and POP:

- SKIP (line 8): We extend each span $[i, j-1]$ in Q to $[i,j]$ by adding an unpaired nucleotide $y_j =$ "." (in the dot-bracket notation) to the right of each substructure in $Q_{i,j-1}$, updating $Q_{i,j}$ as follows:

$$Q_{i,j} \mathrel{+}= Q_{i,j-1} \cdot e^{-\frac{\delta(\mathbf{x},j)}{RT}}$$

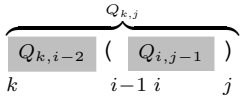
  which can be visualized as



- POP (lines 9–10): If x_{i-1} and x_j are pairable, we combine span $[i, j-1]$ in Q with each combinable "left" span $[k, i-2]$ in Q and update the resulting span $[k,j]$'s partition function as follows:

$$Q_{k,j} \mathrel{+}= Q_{k,i-2} \cdot Q_{i,j-1} \cdot e^{-\frac{\xi(\mathbf{x},i-1,j)}{RT}}.$$

  This means that every substructure in $Q_{i,j-1}$ can be combined with every substructure in $Q_{k,i-2}$ and a base pair $(i-1, j)$ to form one possible substructure in $Q_{k,j}$:



Above we presented a simplified version of our left-to-right LinearPartition algorithm. We have three nested loops, one for j, one for i, and one for k, and each loop takes at most n iterations; therefore, the time complexity *without* beam pruning is $O(n^3)$, which is identical to the classical McCaskill Algorithm (see Fig. 1D). In fact, there is an alternative, bottom-up, interpretation of our left-to-right algorithm that resembles the Nussinov-style recursion of the classical McCaskill Algorithm:

$$Q_{k,j} = Q_{k,j-1} \cdot e^{-\frac{\delta(\mathbf{x},j)}{RT}} + \sum_{k<i\leq j} Q_{k,i-2} \cdot Q_{i,j-1} \cdot e^{-\frac{\xi(\mathbf{x},i-1,j)}{RT}}$$

However, unlike the classical bottom-up McCaskill algorithm, our left-to-right dynamic programming, inspired by LinearFold, makes it possible to further apply the beam pruning heuristic to achieve linear runtime in practice. The main idea is, at each step j, among all possible spans $[i,j]$ that ends at j (with $i = 1 \ldots j$), we only keep the top b most promising candidates (ranked by their partition functions $Q_{i,j}$). where b is the beam size. With such beam pruning, we reduce the number of states from $O(n^2)$ to $O(nb)$, and the runtime from $O(n^3)$ to $O(nb^2)$. For details of the efficient implementation and runtime analysis, please refer to Supplementary Information Section A. Note b is a user adjustable constant (b =100 by default).

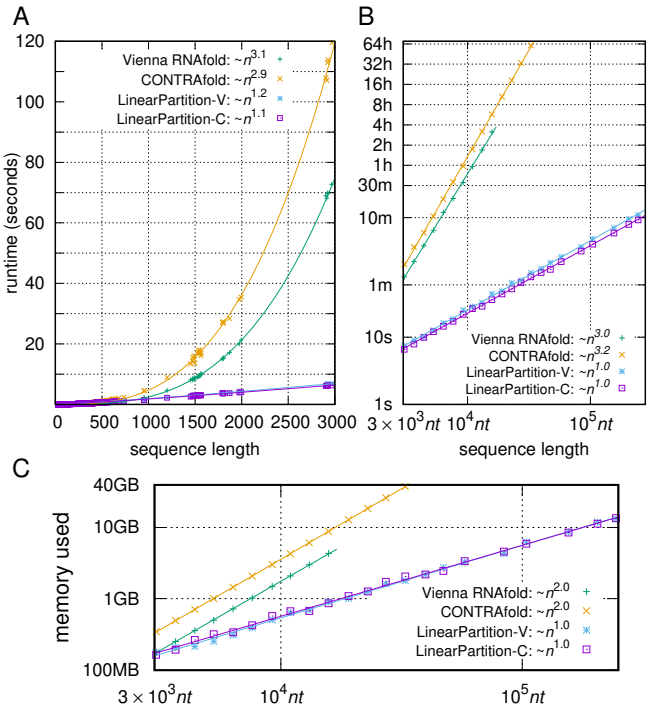


Fig. 4. Total runtime and memory usage of computing both the partition function and base pairing probabilities. **A**: runtime comparisons on the ArchiveII dataset; the curve-fittings were log-log in gnuplot with $n > 10^3$. **B**: runtime comparisons on the RNAcentral dataset (log scale). The partition function computation takes about half of the total time shown here. **C**: Memory usage comparisons on the RNAcentral dataset (log scale).

B. Efficiency and Scalability. We present two versions of LinearPartition: *LinearPartition-V* using thermodynamic parameters [37–39] as implemented in Vienna RNAfold [35], and *LinearPartition-C* using the machine learning-based parameters from CONTRAfold [26]. We run all evaluations on a Linux machine, with 2.90GHz Intel Core i9-7920X CPU and 64G memory.

Figure 4 compares the efficiency and scalability between the 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 (MEA prediction in CONTRAfold, and centroid prediction and visualization in Vienna RNAfold) which are by default enabled. Figure 4A shows that both LinearPartition-V and LinearPartition-C scale almost linearly with sequence length n. The runtime deviation from exact linearity is due to the relatively short sequence lengths in the ArchiveII dataset, which contains a set of sequences with well-determined structures [40]. Figure 4A also confirms that the baselines scale cubically and the $O(n^3)$ runtimes are substantially slower than LinearPartition on long sequences. For the *H. pylori* 23S rRNA sequence (2,968 *nt*, the longest in ArchiveII), both versions of LinearPartition take only 6 seconds, while Vienna RNAfold takes 73 seconds, and CONTRAfold almost 120 seconds.

We also notice that both Vienna RNAfold and CONTRAfold have limitations on even longer sequences. Vienna RNAfold scales the magnitude of the partition function using a constant estimated from the minimum free energy of the given sequence to avoid overflow, but overflows still occur on long sequences. For example, it overflows on the 19,071 *nt* sequence in the sampled RNAcentral dataset. CONTRAfold stores the logarithm of the partition function to solve the

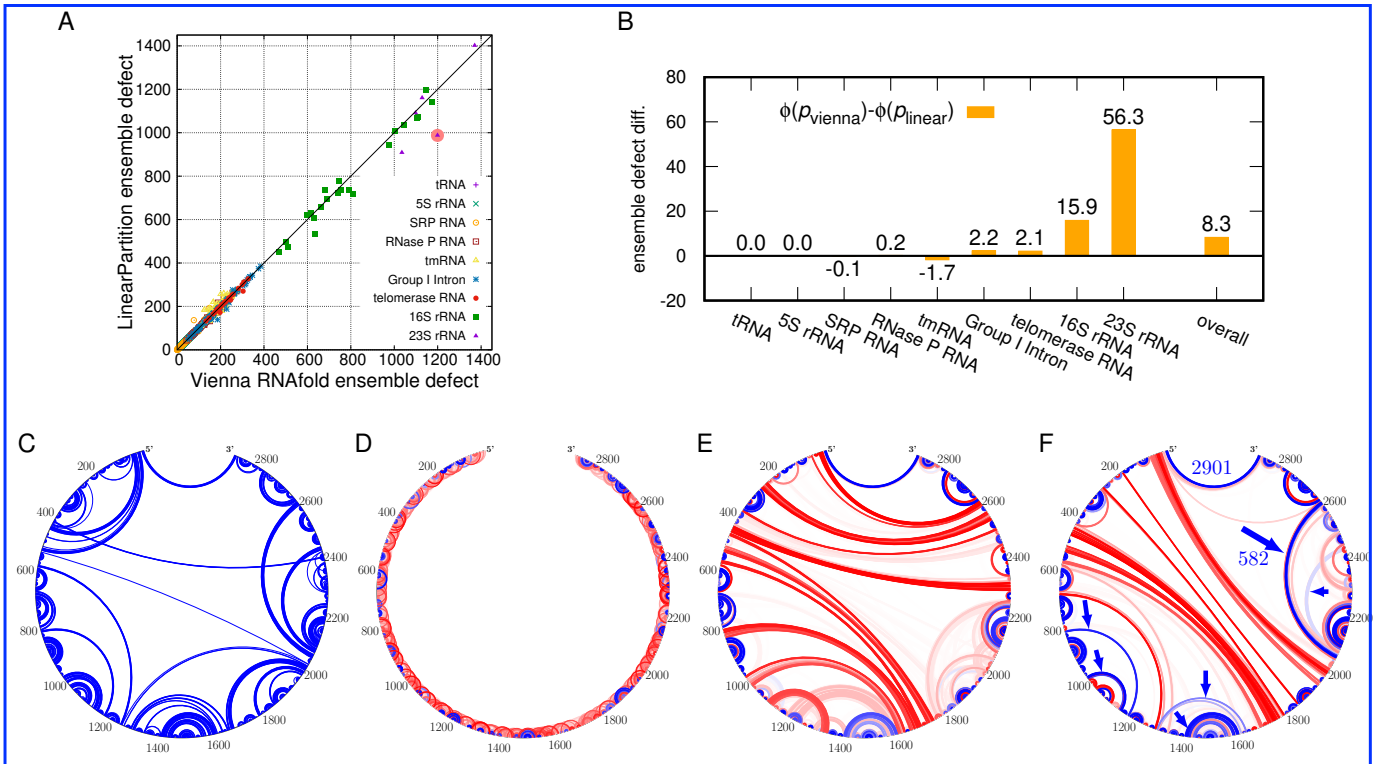


Fig. 5. **A**: Ensemble defect (expected number of incorrectly predicted nucleotides; lower is better) comparison between Vienna RNAfold and LinearPartition on the ArchiveII dataset. **B**: Ensemble defect difference for each family. LinearPartition has lower ensemble defects for longer families: on average 56.3 less incorrectly predicted nucleotides on 23S rRNA and 8.3 less over all families. **C–F**: An example of *E. coli* 23S rRNA (shaded point in **A**). **C**: circular plot of the ground truth. **D–F**: base pair probabilities from Vienna RNAplfold (with default window size 70), RNAfold and LinearPartition, respectively; blue and red indicate correct and incorrect base pairs, and color darkness represents probabilities. See Figs. 6 and SI 4 for another example from Group I Intron.

overflow issue, but cannot run on sequences longer than 32,767 *nt* due to using `unsigned short` to index sequence positions. LinearPartition, like CONTRAfold, performs computations in the log-space, but can run on all sequences in the RNAcentral dataset. Figure 4B compares the runtime of four systems on a sampled subset of RNAcentral dataset. It shows that only LinearPartition can finish all the examples, and on longer sequences the runtime of LinearPartition is exactly linear. Comparing the runtimes of the 15,780 *nt* sequence, the longest example shown for Vienna RNAfold in Figure 4B, Vienna RNAfold takes more than 3 hours and LinearPartition-V only takes 44.1 seconds (256× speedup). Note that Vienna RNAfold may not overflow on some longer sequences, where LinearPartition-V should enjoy an even 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 (2,771× speedup). Even for the longest sequence in RNAcentral (Homo Sapiens Transcript NONHSAT168677.1 with length 244,296 *nt*[41]), LinearPartition-V finishes in 10.9 minutes and LinearPartition-C in 9.2 minutes.

Figure 4C compares the memory usage on RNAcentral-sampled sequences. It confirms that both Vienna RNAfold and CONTRAfold use $O(n^2)$ memory space, while LinearPartition uses $O(n)$ space.

Now that we have established the linearity of LinearPartition, in the next two subsections, we move on to the quality of its output, i.e., the resulting Boltzmann distribution and base pairing probabilities. We first study the correlation between the Boltzmann distribution and the ground truth structures in Section C, and then use base pairing probabilities for downstream structure predictions in Section D.

C. Correlation with Ground Truth Structures. We use the concept of *ensemble defect*[42] to represent the quality of the Boltzmann distribution. It is the expected number of incorrectly predicted nucleotides over the whole ensemble at equilibrium, and formally, for a sequence $\mathbf{x}$ and its ground-truth structure $\mathbf{y}^*$, the ensemble defect is

$$\Phi(\mathbf{x}, \mathbf{y}^*) = \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} p(\mathbf{y}) \cdot d(\mathbf{y}, \mathbf{y}^*) \qquad [4]$$

where $p(\mathbf{y})$ is the probability of structure $\mathbf{y}$ among all possible structures $\mathcal{Y}(\mathbf{x})$, and $d(\mathbf{y}, \mathbf{y}^*)$ is the distance between $\mathbf{y}$ and $\mathbf{y}^*$:

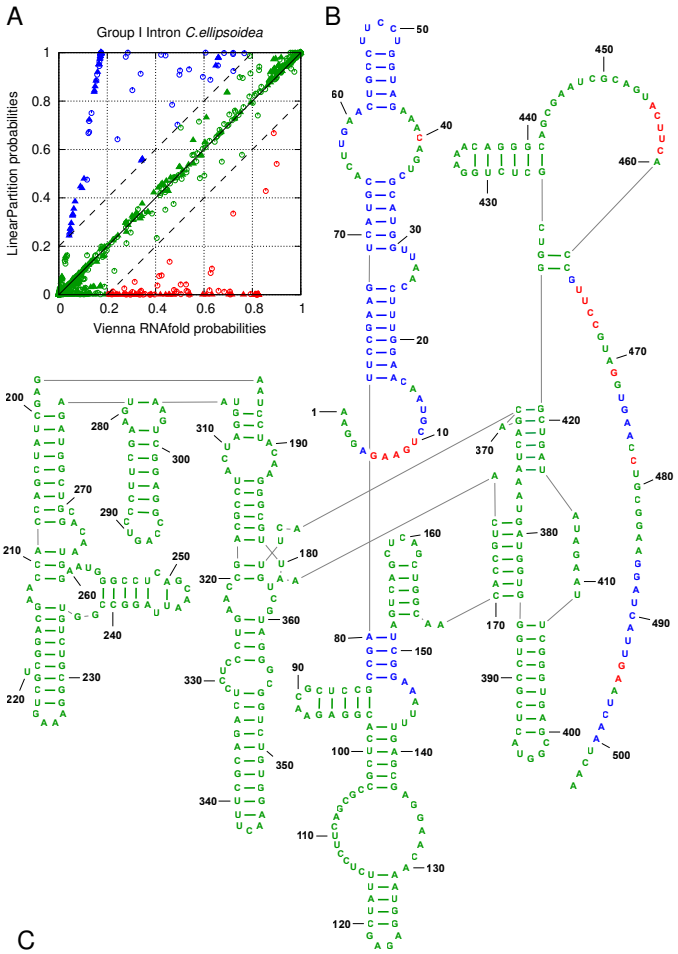
$$d(\mathbf{y}, \mathbf{y}^*) = |\mathbf{x}| - |\text{pairs}(\mathbf{y}) \cap \text{pairs}(\mathbf{y}^*)| - |\text{unpaired}(\mathbf{y}) \cap \text{unpaired}(\mathbf{y}^*)|$$

where $\text{pairs}(\mathbf{y}^*)$ is the set of base pairs in $\mathbf{y}^*$, and $\text{unpaired}(\mathbf{y}^*)$ is the set of unpaired bases in $\mathbf{y}^*$. The naïve calculation of Eq. 4 requires enumerating all possible (exponentially many) structures in the ensemble, but by plugging $d(\mathbf{y}, \mathbf{y}^*)$ into Eq. 4 we have[42]

$$\Phi(\mathbf{x}, \mathbf{y}^*) = |\mathbf{x}| - 2 \sum_{(i,j) \in \text{pairs}(\mathbf{y}^*)} p_{i,j} - \sum_{j \in \text{unpaired}(\mathbf{y}^*)} q_j \qquad [5]$$

where $p_{i,j}$ is the probability of i pairing with j, and q_j is the probability of j being unpaired ($q_j = 1 - \sum p_{i,j}$). This means we can now use base pairing probabilities to compute the ensemble defect.

Figure 5A–B employs ensemble defect to measure the average number and ratio of incorrectly predicted nucleotides over the whole ensemble (lower is better). Vienna RNAfold and LinearPartition have similar ensemble defects for short sequences, but LinearPartition has lower ensemble defects for longer sequences, esp. 16S and 23S rRNAs; in other words, LinearPartition's ensemble has less expected



C

	total		correct	
$p_{\text{linear}} - p_{\text{vienna}} > 0.2$	55 ▲	40 ○	23 ▲	37 ○
$\lvert p_{\text{linear}} - p_{\text{vienna}} \rvert \leq 0.2$	126,645 ▲	420 ○	111 ▲	180 ○
$p_{\text{linear}} - p_{\text{vienna}} < -0.2$	56 ▲	44 ○	0 ▲	19 ○

Fig. 6. **A–C**: An example of *C. ellipsoidea* Group I Intron. **A**: solid triangles (▲ ▲ ▲) stand for base pairing probabilities and unfilled circles (○ ○ ○) stand for single-stranded probabilities. blue: $p_{\text{linear}} - p_{\text{vienna}} > 0.2$; green: $|p_{\text{linear}} - p_{\text{vienna}}| \leq 0.2$; red: $p_{\text{linear}} - p_{\text{vienna}} < -0.2$; **B**: ground truth structure colored with the above scheme; **C**: statistics of this example. "total" columns are the total numbers of triangles and circles with different colors in **A**, while "correct" columns are the corresponding numbers in the ground-truth structure in **B**, which is better correlated with LinearPartition's probabilities than Vienna RNAfold's (23 blue pairs and 0 red pairs).

number of incorrectly predicted nucleotides (or higher number of correctly predicted nucleotides, i.e., better correlation with ground-truth structure). In particular, on 16S and 23S rRNAs, LinearPartition has on average 15.9 and 56.3 more correctly predicted nucleotides than Vienna RNAfold, and on average 8.3 more correctly predicted nucleotides over all families.

This finding also implies that LinearPartition's base pairing probabilities are on average higher than Vienna RNAfold's for ground-truth base pairs, and on average lower for incorrect base pairs. We use two concrete examples to illustrate this. First, we plot the ground truth structure of *E. coli* 23S rRNA (2,904*nt*) in Figure 5C, and then plot the predicted base pairing probabilities from the local folding tool Vienna RNAplfold (with default window size 70), Vienna RNAfold, and LinearPartition in Figure 5D–F, respectively. We can see that local folding can only produce local pairing probabilities, and our Vienna RNAfold misses most of the long-distance pairs from the ground truth (except the 5'-3' helix), and includes many incorrect long-distance pairings (shown in red). By contrast, LinearPartition successfully predicts many long-distance pairings that RNAfold misses, the longest being 582*nt* apart (shown with arrows). Indeed, the ensemble defect of this example confirms that LinearPartition's ensemble distribution has on average 211.4 more correctly predicted nucleotides (over 2,904*nt*, or 7.3%) than RNAfold's.

As the second example, we use *C. ellipsoidea* Group I Intron (504 *nt*). In Figure 6A, we plot the base pairs (in triangle) and unpaired bases (in circle) with Vienna RNAfold probability on x-axis and LinearPartition probability on y-axis. We color the circles and triangles in blue where LinearPartition gives 0.2 higher probability than Vienna RNAfold (top left region), the opposite ones (bottom right region) in red, and the remainder (diagonal region, with probability changes less than 0.2) in green. Then in Figure 6B, we visualize the ground truth structure[43] and color the bases as in Figure 6A. We observe that the majority of bases are in green, indicating that Vienna RNAfold and LinearPartition agree with for a majority of the structure features. But the blue helices near 5'-end and between [80,83] and [148,151] indicate that LinearPartition favors these correct substructures by giving them higher probabilities than Vienna RNAfold. We also notice that all red features (where Vienna RNAfold does better than LinearPartition) are unpaired bases. This example shows that although LinearPartition gives different probabilities compared with Vienna RNAfold, it is likely that LinearPartition prediction structure is closer to the ground truth structure. The ensemble defect of this example also confirms that LinearPartition has on average 47.1 more correctly predicted nucleotides (out of 504*nt*, or 9.3%) than RNAfold.

Figure 6C gives the statistics of this example. We can see the green triangles in Figure 6A, which denote similar base pairing probabilities between Vienna RNAfold and LinearPartition, are the vast majority. The total number of blue triangles, for which LinearPartition gives higher base pairing probabilities, is 55, and among them 23 base pairs (41.8%) 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 higher probabilities to more ground truth unpaired bases: there are 40 blue circles, among which 37 (92.5%) are unpaired in the ground truth structure, while only 19 out of the 44 red circles (43.2%) are in the ground truth structure. See also Fig. SI 4 for another view of this example in the style of Fig. 5C–F using circular plots.

D. Accuracy of Downstream Structure Predictions. An important application of the partition function is to improve structure prediction accuracy (over MFE) using base pairing probabilities. Here we use two such "downstream prediction" methods, MEA[26] and ThreshKnot[29] which is a thresholded version of ProbKnot[28], and compare their results using base pairing probabilities from $O(n^3)$-time baselines and our $O(n)$-time LinearPartition. We use Positive Predictive Value (PPV, the fraction of predicted pairs in the known structure, a.k.a. precision) and sensitivity (the fraction of known pairs predicted, a.k.a. recall) as accuracy measurements for each family, and get overall accuracy be averaging over families. When scoring accuracy, we allow base pairs to differ by one nucleotide in position[37]. We compare Vienna RNAfold and LinearPartition-V on the ArchiveII dataset in the main text, and provide the CONTRAfold vs. LinearPartition-C comparisons in the Supporting Information Figs. SI 5–SI 6.

Figure 7A shows that MEA predictions (Vienna RNAfold + MEA and LinearPartition + MEA) are more accurate than MFE ones (Vienna RNAfold MFE and LinearFold-V), but more importantly, Lin-

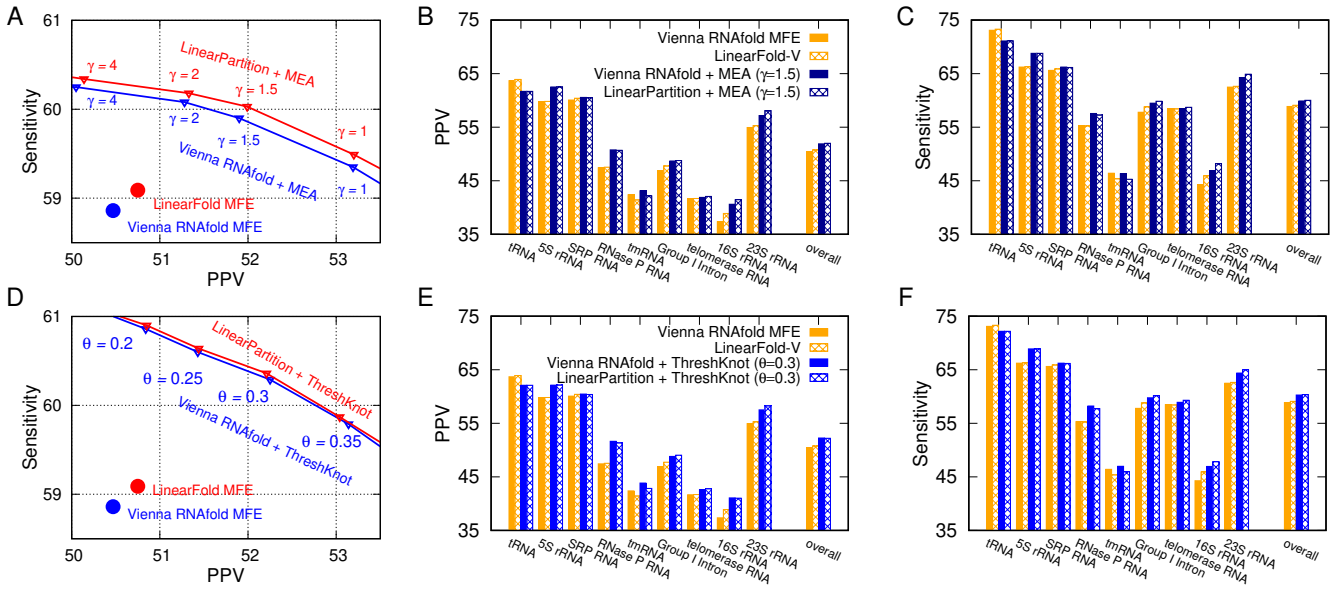


Fig. 7. Accuracy of downstream structure predictions (MEA and ThreshKnot) using base pairing probabilities from Vienna RNAfold and LinearPartition on the ArchiveII dataset. **A**: Overall PPV-Sensitivity tradeoff of MFE (single point) and MEA with varying γ (which can be tuned for higher sensitivity or PPV by adjusting γ). **B** and **C**: PPV and Sensitivity comparisons of MEA structures for each family. **D–F**: same as **A–C**, but using ThreshKnot predictions instead of MEA. We conclude that MEA predictions based on LinearPartition-V are consistently better in both PPV and Sensitivity than those based on Vienna RNAfold for all γ's, while ThreshKnot predictions based on those two are almost identical for all θ's.

earPartition + MEA consistently outperforms Vienna RNAfold + MEA in both PPV and sensitivity with the same γ, a hyperparameter that balances PPV and sensitivity in the MEA algorithm.

Figures 7B–C detail the per-family PPV and sensitivity, respectively, for MFE and MEA ($\gamma = 1.5$) results from Figure 7A. LinearPartition + MEA has similar PPV and sensitivity as Vienna RNAfold + MEA on short families (tRNA, 5S rRNA and SRP), but interestingly, is more accurate on longer families, especially the two longest ones, 16S rRNA (+0.86 on PPV and +1.29 on sensitivity) and 23S rRNA (+0.88 on PPV and +0.62 on sensitivity).

ProbKnot is another downstream prediction method that is simpler and faster than MEA; it assembles base pairs with reciprocal highest pairing probabilities. Recently, we demonstrated ThreshKnot[29], a simple thresholded version of ProbKnot, leads to more accurate predictions that outperform MEA by filtering out unlikely pairs, i.e., those whose probabilities fall under a given threshold θ.

Shown in Fig. 7D, LinearPartition + ThreshKnot is almost identical in overall accuracy to Vienna RNAfold + ThreshKnot at all θ's, and is slightly better than the latter on long families (+0.24 on PPV and +0.38 on sensitivity for Group I Intron, +0.12 and +0.37 for telomerase RNA, and +0.74 and +0.62 for 23S rRNA) (see Figs. 7E–F). We also performed a two-tailed permutation test to test the statistical significance, and observed that on tmRNA, both MEA and ThreshKnot structures of LinearPartition are significantly worse ($p < 0.01$) than their Vienna RNAfold-based counterparts in both PPV and Sensitivity.

Figs. SI 5–SI 6 show similar comparisons between CONTRAfold and LinearPartition-C using MEA and ThreshKnot prediction, with similar results to Fig. 7, i.e., downstream structure prediction using LinearPartition-C is as accurate as using CONTRAfold, and (sometimes significantly) more accurate on longer families.

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

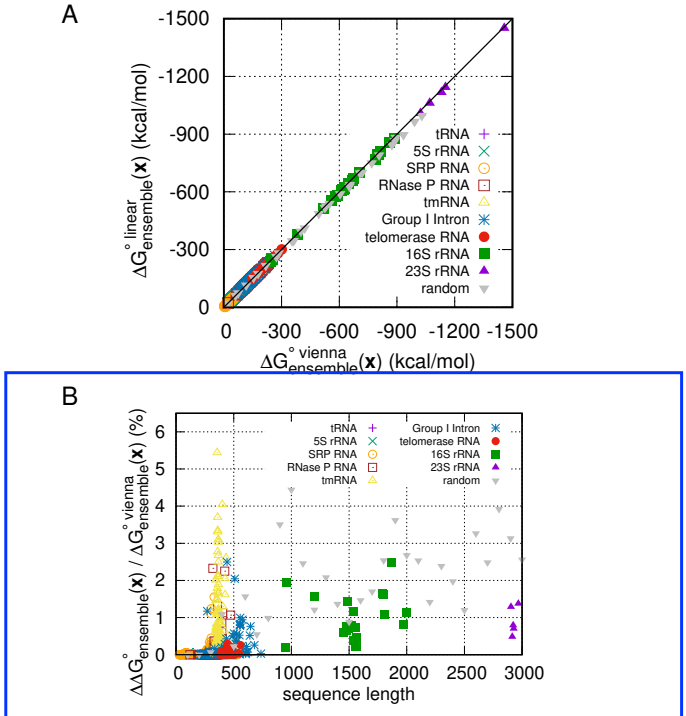


Fig. 8. Approximation quality of partition function on ArchiveII dataset and random sequences. **A**: the x and y axes are ensemble folding free energy changes $\Delta G^\circ_{\text{ensemble}}(\mathbf{x})$ of Vienna RNAfold and LinearPartition, respectively. **B**: the percentage of ensemble folding free energy changes between Vienna RNAfold and LinearPartition $\frac{\Delta G^{\circ\,\text{vienna}}_{\text{ensemble}}(\mathbf{x}) - \Delta G^{\circ\,\text{linear}}_{\text{ensemble}}(\mathbf{x})}{\Delta G^{\circ\,\text{linear}}_{\text{ensemble}}(\mathbf{x})}$.

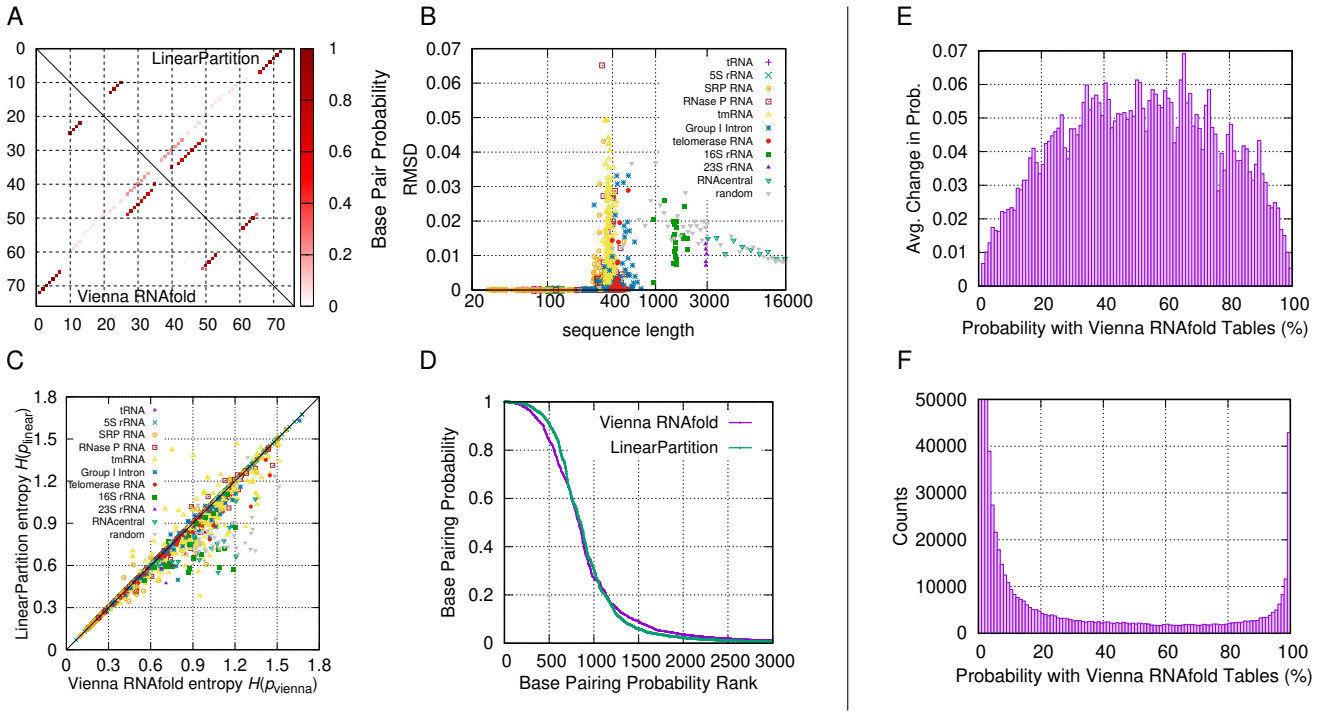


Fig. 9. Comparison of base pairing probabilities from Vienna RNAfold and LinearPartition. **A**: LinearPartition (upper triangle) and Vienna RNAfold (lower triangle) result in identical base pairing probability matrix for *E. coli* tRNA^{Gly}. **B**: the root-mean-square deviation, $\text{RMSD}(p_{\text{vienna}}, p_{\text{linear}})$, 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**: average positional structural entropy $H(p)$ comparison. **D**: LinearPartition starts higher and finishes lower than Vienna RNAfold in a sorted probability curve for *E. coli* 23S rRNA. **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.

We first measure the approximation quality of the partition function calculation, and specifically, we measure the ensemble folding free energy change (also known as "free energy of the ensemble") which reflects the size of the partition function,

$$\Delta G^{\circ}_{\text{ensemble}}(\mathbf{x}) = -RT \log Q(\mathbf{x}).$$

Figure 8 shows that the LinearPartition estimate for the ensemble folding free energy change is close to the Vienna RNAfold estimate on the ArchiveII dataset and randomly generated RNA sequences. The similarity shows that little magnitude of the partition function is lost by the beam pruning. For short families, free energy of ensembles between LinearPartition and Vienna RNAfold are almost the same. For 16S and 23S rRNA sequences and long random sequences (longer than 900 nucleotides), LinearPartition gives a lower magnitude ensemble free energy change, but the difference is smaller than 19.5 kcal/mol for 16S rRNA, 14.6 kcal/mol for 23S rRNA and 36.7 kcal/mol for random sequences. The maximum difference for random sequence is bigger than natural sequences (by 17.2 kcal/mol). This likely reflects the fact that random sequences tend to fold less selectively to probable structures [44], and the beam is therefore pruning structures in random that would contribute to the overall folding stability.

Next, we measure the quality of base pairing probability approximation using root-mean-square deviation (RMSD) between two probability matrices p and p' (i.e., p_{vienna} and p_{linear}) over the set of all possible Watson-Crick and wobble base pairs on a sequence $\mathbf{x}$. More formally, we define

$$\text{pairings}(\mathbf{x}) = \{1 \le i < j \le |\mathbf{x}| \;\big|\; \mathbf{x}_i\mathbf{x}_j \in \{\text{CG, GC, AU, UA, GU, UG}\}, j-i>3\}$$

and:

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

Figures 9A and B confirm that our LinearPartition algorithm (with default beam size 100) can indeed approximate the base pairing probability matrix reasonably well. Figure 9A shows the heatmap of probability matrices for *E. coli* tRNA^{Gly}. Vienna RNAfold (lower triangle) and LinearPartition (upper triangle) yield identical matrices (i.e., RMSD $= 0$). Figure 9B shows that the RMSD of each sequence in ArchiveII and RNAcentral datasets, and randomly generated artificial RNA sequences, is relatively small. The highest deviation is 0.065 for *A. truei* RNase P RNA, 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, the 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 close to 0 (RMSD$< 10^{-5}$) for short ones, then becomes bigger around length 500 and decreases after that, but for most cases their RMSD's are slightly larger than the natural sequences. This indicates that the approximation quality

is relatively better for natural sequences. For RNAcentral-sampled sequences, RMSD's are all small and around 0.01.

We assume LinearPartition base pairing probabilities distribution is shifted to higher probability because it filters out states with lower partition function. We measure this by using average positional structural entropy $H(p)$, which is the average of positional structural entropy $H_2(i)$ for each nucleotide i[45,46]:

$$H(p) = \frac{1}{n}\sum_{i=1}^{n} H_2(i) = \frac{1}{n}\sum_{i=1}^{n}(-\sum_{j=0}^{n} p_{i,j}\log_2 p_{i,j}) \\ = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=0}^{n} p_{i,j}\log_2 p_{i,j} \quad [6]$$

where p is the base pairing probability matrix, $p_{i,j}$ is the probability of nucleotide i paired with j when $j \neq 0$, and $p_{i,0}$ is the probability of nucleotide i being unpaired. n is the sequence length. The lower entropy indicates that the distribution is dominated by fewer base pairing probabilities. Figure 9D confirms LinearPartition distribution shifted to higher probabilities (lower average positional structural entropy) than Vienna RNAfold for most sequences.

We also use *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 9C shows the LinearPartition distribution curve starts higher and finishes lower, confirming the distribution shifts to higher probabilities.

Figures 9E and F follow a previous analysis method[47] to estimate the approximation quality with 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 9E 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 9F we illustrate the counts in each bin based on Vienna RNAfold base pairing probabilities. We can see that most base pairs have low probabilities (near 0) or very high probabilities (near 1). Combine Figures 9E 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 7 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 higher 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 base pairs with probabilities close to 0.

F. Adjustable Beam Size. Beam size in LinearPartition is a user adjustable hyperparameter controlling beam prune, and it balances 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 is infinite (no beam prune). Figure 10A 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 average RMSD is lower than 0.035 over all ArchiveII sequences. With default beam size ($b = 100$) the average RMSD is lower than 0.005. With a larger beam size, $b = 500$, the average RMSD decreases to almost 0.

Beam size also has impact on PPV and Sensitivity. Figure 10B 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 $b = 100$. Therefore, we choose beam size 100 as the default beam size. Figures 10C and D present this impact for two selected families. Figure 10C 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 10D 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[48] to reduce runtime from $O(nb^2)$ to $O(nb\log b)$ without 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 we applied k-best parsing here, LinearPartition would sum the partition function of only these top-b states instead of the partition function of b^2 states. This change would introduce a larger approximation error, especially when the differences of partition function between the top-b states and the following states near the pruning boundary are small. Therefore, in LinearPartition we do not use k-best parsing as in LinearFold, and the runtime is $O(nb^2)$ instead of $O(b\log b)$.

3. Discussion

A. Summary. Classical partition function and base pairing probabilities calculations are performed in many studies of RNA sequences, but their application has been limited to long sequences (such as full length mRNA) because of the limitation imposed by their cubic scaling. To address this issue, we present LinearPartition, an algorithm that can dramatically reduce runtime of partition function and base pairing probabilities calculations. 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 (Figure 4).

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 (Figure 7).

3. The approximation quality of LinearPartition is good. Although filtering out some structures, the ensemble free energy change of LinearPartition is either the same or only slightly smaller than Vienna RNAfold, e.g., the largest fraction of total free energy change is 2.5% in the ArchiveII dataset (Figure 8). Additionally, RMSD of base pairing probabilities between LinearPartition and Vienna RNAfold is small, e.g., the largest RMSD in the ArchiveII dataset is 0.065 (Figure 9).

4. With increasing beam size, 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 with an increase beam size (Figure 10).

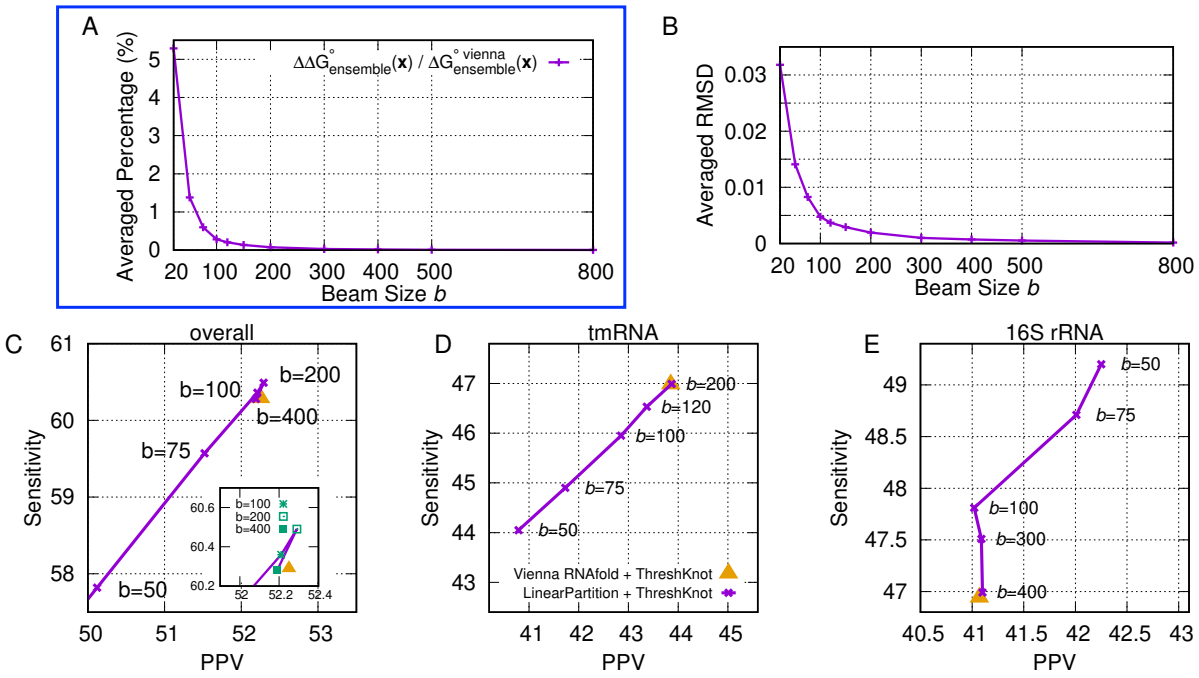


Fig. 10. Impact of beam size. **A**: percentage of ensemble folding free energy changes with beam size. **B**: RMSD changes with beam size. **C**: overall PPV and Sensitivity change with beam size. **D–E**: tmRNA and 16S rRNA PPV and Sensitivity change with beam size, respectively. Note that the yellow triangles in **B–D** are identical to ThreshKnot using the *exact* version of LinearPartition (with $b = \infty$).

B. Extensions. Our algorithm has several potential extensions.

1. Bimolecular and multistrand base pairing probabilities and accessibility could be accelerated and improved. Many ncRNAs function by interacting with other RNA sequences by base pairing. Existing methods and tools for calculating two-strand (bimolecular) or multi-strand folding partition functions and base pairing probability matrices [49–51] suffer from slowness, limiting the 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. 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-designed constrains. Compared with solving the 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, such as FastIPknot or FastDotKnot. With these promising results of LinearPartition, we believe FastIPknot (and FastDotKnot) might be as accurate as, if not more accurate than, their original $O(n^3)$ versions.

Methods

Datasets. We use sequences from two datasets, ArchiveII and RNAcentral. The archiveII dataset (available in http://rna.urmc.rochester.edu/pub/archiveII.tar.gz) is a diverse set with 3,857 RNA sequences and their secondary structures. It is first curated in the 1990s to contain sequences with structures that were well-determined by comparative sequence analysis [37] and updated later with additional structures [40]. We remove 957 sequences that appear both in the ArchiveII and the S-Processed datasets [52], because CONTRAfold uses S-Processed for training. We also remove all 11 Group II Intron sequences because there are so few instances of these that are available electronically. Additionally, we removed 30 sequences in the tmRNA family because the annotated structure for each of these sequences contains fewer than 4 pseudoknots, which suggests the structures are incomplete. These preprocessing steps lead to a subset of ArchiveII with 2,859 reliable secondary structure examples distributed in 9 families. See SI 1 for the statistics of the sequences we use in the ArchiveII dataset. Moreover, we randomly sampled 22 longer RNA sequences (without known structures) from RNAcentral (https://rnacentral.org/), with sequence lengths ranging from 3,048 *nt* to 244,296 *nt*. For the sampling, we evenly split the range from 3,000 to 244,296 (the longest) into 24 bins by log-scale, and for each bin we randomly select a sequence (there are bins with no sequences).

To show the approximation quality on random RNA sequences, we generated 30 sequences with uniform distribution over {A, C, G, U}. The lengths of these sequences are spaced in 100 nucleotide intervals from 100 to 3,000.

Baseline Software. We use two baseline software packages: (1) Vienna RNAfold (Version 2.4.11) from https://www.tbi.univie.ac.at/RNA/download/sourcecode/2_4_x/ViennaRNA-2.4.11.tar.gz and (2) CONTRAfold (Version 2.0.2) from 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 provide partition function and base pairing probability calculations based on the 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 bug in CONTRAfold by comparing our results to CONTRAfold, which led to overcounting multiloops in the partition function calculation. We corrected the

bug, and all experiments are based on this bug-fixed version of CONTRAfold.

Evaluation Metrics and Significance Test. Due to the uncertainty of base-pair matches existing in comparative analysis and the fact that there is fluctuation in base pairing at equilibrium, we consider a base pair to be correctly predicted if it is also displaced by one nucleotide on a strand [37]. Generally, if a pair (i, j) is in the predicted structure, we consider it a correct prediction if one of (i, j), $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, $(i, j+1)$ is in the ground truth structure.

We use Positive Predictive Value (PPV) and sensitivity as accuracy measurements. Formally, denote $\mathbf{y}$ as the predicted structure and $\mathbf{y}^*$ as the ground truth, we have:

$$\text{PPV} = \frac{\#_{\text{TP}}}{\#_{\text{TP}} + \#_{\text{FP}}} = \frac{|\text{pairs}(\mathbf{y}) \cap \text{pairs}(\mathbf{y}^*)|}{|\text{pairs}(\mathbf{y})|}$$

$$\text{Sensitivity} = \frac{\#_{\text{TP}}}{\#_{\text{TP}} + \#_{\text{FN}}} = \frac{|\text{pairs}(\mathbf{y}) \cap \text{pairs}(\mathbf{y}^*)|}{|\text{pairs}(\mathbf{y}^*)|}$$

where $\#_{\text{TP}}$ is the number of true positives (correctly predicted pairs), $\#_{\text{FP}}$ is the number of false positives (wrong predicted pairs) and $\#_{\text{FN}}$ is the number of false negatives (missing ground truth pairs).

We test statistical significance using a paired, two-sided permutation test [53]. We follow the common practice, choosing $10,000$ as the repetition number and $\alpha = 0.05$ as the significance threshold.

Curve Fitting. We determine the best exponent a for the scaling curve $O(n^a)$ for each data series in Figures 2 and 4. Specifically, we use $f(x) = ax + b$ to fit the log-log plot of those series in Gnuplot; e.g., fitting $\log t_n = a \log n + b$, where t_n is the running time on a sequence of length n, so that $t_n = e^b n^a$. Gnuplot uses the nonlinear least-squares Marquardt-Levenberg algorithm.

Code availability

Our LinearPartition source code can be downloaded from https://github.com/LinearFold/LinearPartition.

Data availability

The data that support the findings of this study are available from the corresponding author upon request.

ACKNOWLEDGMENTS. This work was partially supported by NSF grant IIS-1817231 (L.H.) and NIH grant R01 GM076485 (D.H.M.). We thank Rhiju Das for the early adoption of our software in the EteRNA game.

References

1. S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews Genetics*, 2(12):919–929, 2001.
2. Jennifer A. Doudna and Thomas R. Cech. The chemical repertoire of natural ribozymes. *Nature*, 418(6894):222–228, 2002.
3. Jean Pierre Bachellerie, Jérôme Cavaillé, and Alexander Hüttenhofer. The expanding snoRNA world. *Biochimie*, 84(8):775–790, 2002.
4. Jinwei Zhang and Adrian R Ferré-D'Amaré. New molecular engineering approaches for crystallographic studies of large RNAs. *Current opinion in structural biology*, 26:9—15, June 2014.
5. Huaqun Zhang and Sarah Keane. Advances that facilitate the study of large RNA structure and dynamics by nuclear magnetic resonance spectroscopy. *Wiley Interdisciplinary Reviews: RNA*, 10:e1541, 04 2019.
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. Samuel Coulbourn Flores and Russ B. Altman. Turning limited experimental information into 3d models of RNA. *RNA*, 16(9):1769–1778, 2010.
10. Matthew G. Seetin and David H. Mathews. Automated RNA tertiary structure prediction from secondary structure and low-resolution restraints. *Journal of Computational Chemistry*, 32(10):2232–2244, 2011.
11. 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.
12. J. S. McCaskill. The equilibrium partition function and base pair probabilities for RNA secondary structure. *Biopolymers*, 29:11105–1119, 1990.
13. Sita J. Lange, Daniel Maticzka, Mathias Mohl, Joshua N. Gagnon, Chris M. Brown, and Rolf Backofen. Global or local? predicting secondary structure and accessibility in mRNAs. *Nucleic Acids Research*, 40(12):5215–5226, 2012.
14. Stephan H Bernhart, Ivo L Hofacker, and Peter F Stadler. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–615, 2006.
15. Hisanori Kiryu, Taishin Kin, and Kiyoshi Asai. Rfold: an exact algorithm for computing local base pairing probabilities. *Bioinformatics*, 24(3):367–373, 2008.
16. 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.
17. Pablo Cordero and Rhiju Das. Rich RNA structure landscapes revealed by mutate-and-map analysis. *PLOS Computational Biology*, 11(11), 2015.
18. 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.
19. 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.
20. 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.
21. 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.
22. Z. J. Lu and D. H. Mathews. Efficient siRNA selection using hybridization thermodynamics. *Nucleic Acids Research*, 36:640–647, 2008.
23. 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.
24. 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.
25. B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research*, 31(13):3423–3428, 2003.
26. Chuong Do, Daniel Woods, and Serafim Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, 2006.
27. 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.
28. Stanislav Bellaousov and David H Mathews. Probknot: fast prediction of RNA secondary structure including pseudoknots. *RNA*, 16(10):1870–1880, 2010.
29. Liang Zhang, He Zhang, David H. Mathews, and Liang Huang. ThreshKnot: Thresholded ProbKnot for improved RNA secondary structure prediction. *arXiv preprint 1912.12796* https://arxiv.org/abs/1912.12796, 2019.
30. 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.
31. 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.
32. 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.
33. David H Mathews. Revolutions in RNA secondary structure prediction. *Journal of molecular biology*, 359(3):526–532, 2006.
34. 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.
35. 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.
36. Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL 2010*, page 1077–1086, Uppsala, Sweden, 2010. ACL.
37. 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.
38. 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.
39. Tianbing Xia, John SantaLucia, Mark E. Burkard, Ryszard Kierzek, Susan J. Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with watson-crick base pairs. *Biochemistry*, 37(42):14719–14735, 1998. PMID: 9778347.
40. 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.
41. 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.
42. 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.

43. Jamie J Cannone, Sankar Subramanian, Murray N Schnare, James R Collett, Lisa M D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V Madabusi, Kirsten M Müller, Nupur Pande, Zhidi Shang, Nan Yu, and Robin R Gutell. The comparative RNA web (crw) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BioMed Central Bioinformatics*, 3(2), 2002.
44. Yinghan Fu, Zhenjiang Zech Xu, Zhi J. Lu, Shan Zhao, and David H. Mathews. Discovery of novel ncRNA sequences in multiple genome alignments on the basis of conserved and stable secondary structures. *PLoS One*, 10(6), 2015.
45. Martijn Huynen, Robin Gutell, and Danielle Konings. Assessing the reliability of RNA folding using statistical mechanics. *Journal of Molecular Biology*, 267:1104–1112, 1997.
46. Juan Antonio Garcia-Martin and Peter Clote. RNA thermodynamic structural entropy. *PLoS ONE*, 10(11), 11 2015.
47. 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.
48. Liang Huang and David Chiang. Better k-best parsing. *Proceedings of the Ninth International Workshop on Parsing Technologies*, pages 53–64, 2005.
49. 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.
50. 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. *Nature Structural & Molecular Biology*, 17:83–89, 2009.
51. 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.
52. Mirela Andronescu, Anne Condon, Holger H. Hoos, David H. Mathews, and Kevin P. Murphy. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23:i19–28, 2007.
53. Nima Aghaeepour and Holger H Hoos. Ensemble-based prediction of RNA secondary structures. *BMC Bioinformatics*, 14(139).

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

A. Details of the Efficient Implementation

A.1 Data Structures

In the main text, for simplicity of presentation, Q is described as a hash from span $[i, j]$ to $Q_{i,j}$, but in our actual implementation, to make sure the overall runtime is $O(nb^2)$, we implement Q as an array of n hashes, where each $Q[j]$ is a hash mapping i to $Q[j][i]$ which is conveniently notated as $Q_{i,j}$ in the main text. It is important to note that the first dimension j is the right boundary and the second dimension i is the left boundary of the span $[i, j]$. See the following table for a summary of notations and the corresponding actual implementations.

notations in this paper	actual Python implementation
$Q \leftarrow$ hash()	`Q = [defaultdict(float) for _ in range(n)]`
$Q_{i,j}$	`Q[j][i]`
$[i, j]$ in Q	`i in Q[j]`
for each i such that $[i, j]$ in Q	`for i in Q[j]`
delete $[i, j]$ from Q	`del Q[j][i]`

A.2 Complexity Analysis

In the partition function calculation (inside phase) in Fig. 3, the number of states is $O(nb)$ because each $Q[j]$ contains at most b states ($Q_{i,j}$'s) after pruning. Therefore the space complexity is $O(nb)$. For time complexity, there are three nested loops, the first one (j) with n iterations, the second (i) and the third (k) loops both have $O(b)$ iterations thanks to pruning, so the overall runtime is $O(nb^2)$.

A.3 Outside Partition Function and Base Pairing Probability Calculation

After we compute the partition functions $Q_{i,j}$ on each span $[i, j]$ (known as the "inside partition function"), we also need to compute the complementary function $\widehat{Q}_{i,j}$ for each span known as the "outside partition function" in order to derive the base-pairing probabilities. Unlike the inside phase, this outside partition function is calculated from top down, with $\widehat{Q}_{1,n} = 1$ as the base case.

$$\begin{aligned}\widehat{Q}_{i,j} = &\widehat{Q}_{i,j+1} \cdot e^{-\frac{\delta(\mathbf{x},j+1)}{RT}} \\ &+ \sum_{k<i} \widehat{Q}_{k,j+1} \cdot Q_{k,i-2} \cdot e^{-\frac{\xi(\mathbf{x},i-1,j+1)}{RT}} \\ &+ \sum_{k>j+1} \widehat{Q}_{i,k} \cdot Q_{j+2,k-1} \cdot e^{-\frac{\xi(\mathbf{x},j+1,k)}{RT}}\end{aligned}$$

Note that the second line is only possible when $x_{i-1}x_{j+1}$ can form a base pair (otherwise $e^{-\frac{\xi(\mathbf{x},i-1,j+1)}{RT}} = 0$) and the third line has a constraint that $x_{j+1}x_k$ can form a base pair (otherwise $e^{-\frac{\xi(\mathbf{x},j+1,k)}{RT}} = 0$).

For each (i, j) where $x_i x_j$ can form a base pair, we compute its pairing probability:

$$p_{i,j} = \sum_{k \le i} \widehat{Q}_{k,j} \cdot Q_{k,i-1} \cdot e^{-\frac{\xi(\mathbf{x},i,j)}{RT}} \cdot Q_{i+1,j-1}$$

The whole "outside" computation takes $O(n^3)$ without pruning, but also $O(nb^2)$ with beam pruning. See Fig. SI 2 for the pseudocode to compute the outside partition function and base pairing probabilities.

B. Supporting Tables and Figures

function BEAMPRUNE(Q, j, b)
  $candidates \leftarrow$ hash() ▷ hash table: from candidates i to score
  for each i such that $[i, j]$ in Q **do**
    $candidates[i] \leftarrow Q_{1,i-1} \cdot Q_{i,j}$ ▷ like LinearFold, use $Q_{1,i-1}$ as prefix score
  $candidates \leftarrow$ SELECTTOPB($candidates, b$) ▷ select top-b states by score
  for each i such that $[i, j]$ in Q **do**
    if key i not in $candidates$ **then**
      delete $[i, j]$ from Q ▷ prune low-scoring states

Fig. SI 1. The BEAMPRUNE function from the Pseudocode of our main algorithm (Fig. 3).

function BASEPAIRINGPROBS($\mathbf{x}, Q$) ▷ outside calculation
  $n \leftarrow$ length of $\mathbf{x}$
  $\widehat{Q} \leftarrow$ hash() ▷ hash table: from span $[i, j]$ to $\widehat{Q}_{i,j}$: outside partition function
  $p \leftarrow$ hash() ▷ hash table: from span $[i, j]$ to $p_{i,j}$: base-pairing probs
  $\widehat{Q}_{1,n} \leftarrow 1$ ▷ base case
  for $j = n$ **downto** 1 **do**
    for each i such that $[i,\ j-1]$ in Q **do**
      $\widehat{Q}_{i,j-1}$ += $\widehat{Q}_{i,j} \cdot e^{-\frac{\delta(\mathbf{x},j)}{RT}}$ ▷ SKIP
      if $x_{i-1}x_j$ in {AU, UA, CG, GC, GU, UG} **then**
        for each k such that $[k,\ i-2]$ in Q **do**
          $\widehat{Q}_{k,i-2}$ += $\widehat{Q}_{k,j} \cdot Q_{i,j-1} \cdot e^{-\frac{\xi(\mathbf{x},i-1,j)}{RT}}$ ▷ POP: left
          $\widehat{Q}_{i,j-1}$ += $\widehat{Q}_{k,j} \cdot Q_{k,i-2} \cdot e^{-\frac{\xi(\mathbf{x},i-1,j)}{RT}}$ ▷ POP: right
          $p_{i-1,\,j}$ += $\dfrac{\widehat{Q}_{k,j} \cdot Q_{k,i-2} \cdot e^{-\frac{\xi(\mathbf{x},i-1,j)}{RT}} \cdot Q_{i,j-1}}{Q_{1,n}}$ ▷ accumulate base pairing probs
  return p ▷ return the (sparse) base-pairing probability matrix

Fig. SI 2. Outside partition function and base pairing probabilities calculation for a simplified version of the LinearPartition. Q is the (inside) partition function calculated by the pseudocode in Fig. 3, and $\widehat{Q}$ is the outside partition function. The actual algorithm using the Turner model is in our GitHub codebase.

	# of seqs		length		
Family	total	used	avg	max	min
tRNA	557	74	77.3	88	58
5S rRNA	1,283	1,125	118.8	135	102
SRP RNA	928	886	186.1	533	28
RNase P RNA	454	182	344.1	486	120
tmRNA	462	432	369.1	433	307
Group I Intron	98	96	424.9	736	210
Group II Intron	11	0	-	-	-
telomerase RNA	37	37	444.6	559	382
16S rRNA	22	22	1,547.9	1995	950
23S rRNA	5	5	2,927.4	2968	2904
Overall	3,846	2,859	221.1	2968	28

Table SI 1. Statistics of the sequences in the ArchiveII dataset used in this work.

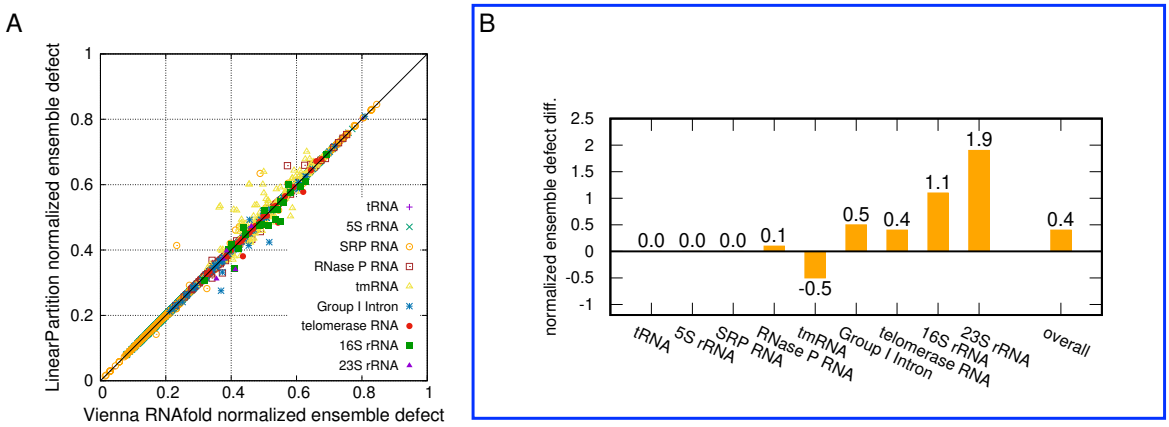


Fig. SI 3. Normalized ensemble defect comparison of Vienna RNAfold and LinearPartition-V on the ArchiveII dataset. **A**: ensemble defects are normalized by their sequence length. The trend is similar as **A**, but the deviations for tmRNAs are more apparent. **B**: normalized ensemble defect difference for each family; for longer families, e.g., Group I Intron, telomerase RNA, 16S and 23S rRNA, LinearPartition has lower normalized ensemble defect differences.

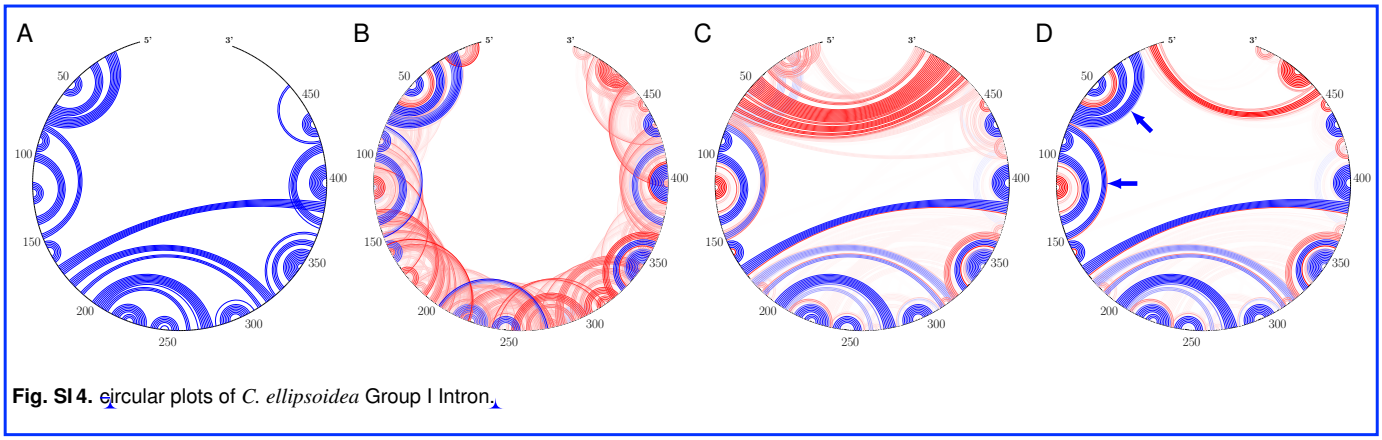


Fig. SI 4. circular plots of *C. ellipsoidea* Group I Intron.

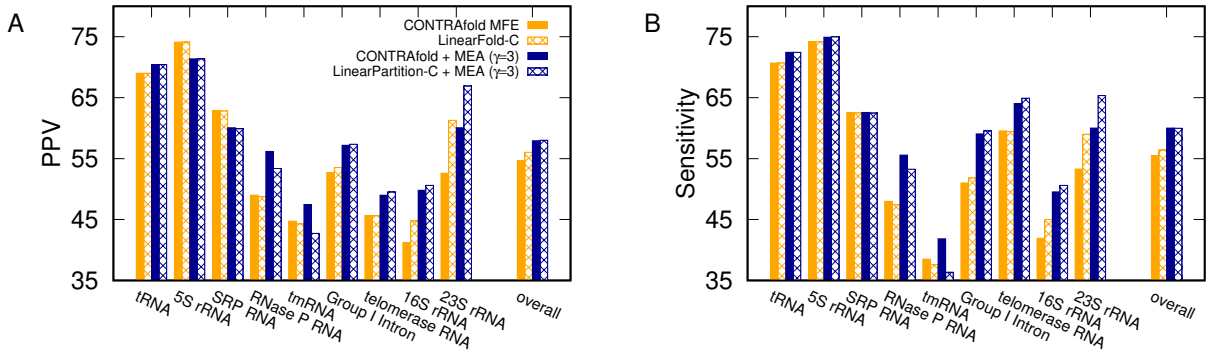


Fig. SI 5. Accuracy comparison of MEA structures ($\gamma = 3$) between CONTRAfold and LinearPartition-C on the ArchiveII dataset. γ is the hyperparameter balances PPV and Sensitivity. Note that LinearPartition-C + MEA is significantly worse than CONTRAfold + MEA on two families in both PPV and Sensitivity, tmRNA and RNase P RNA ($p < 0.01$).

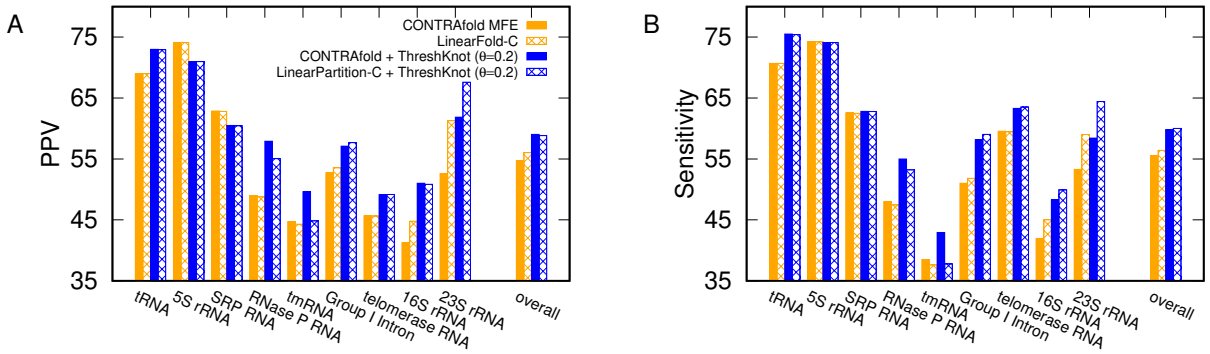


Fig. SI 6. Accuracy comparison of ThreshKnot structure ($\theta = 0.2$) between CONTRAfold and LinearPartition-C on ArchiveII dataset. θ is the hyperparameter that balances PPV and Sensitivity. Note that LinearPartition-C + ThreshKnot is significantly worse than CONTRAfold + ThreshKnot on two families in both PPV and Sensitivity, tmRNA and RNase P RNA ($p < 0.01$), and significantly better on three longer families in Sensitivity, Group I Intron ($p < 0.01$), telomerase RNA and 16S rRNA ($0.01 \leq p < 0.05$).

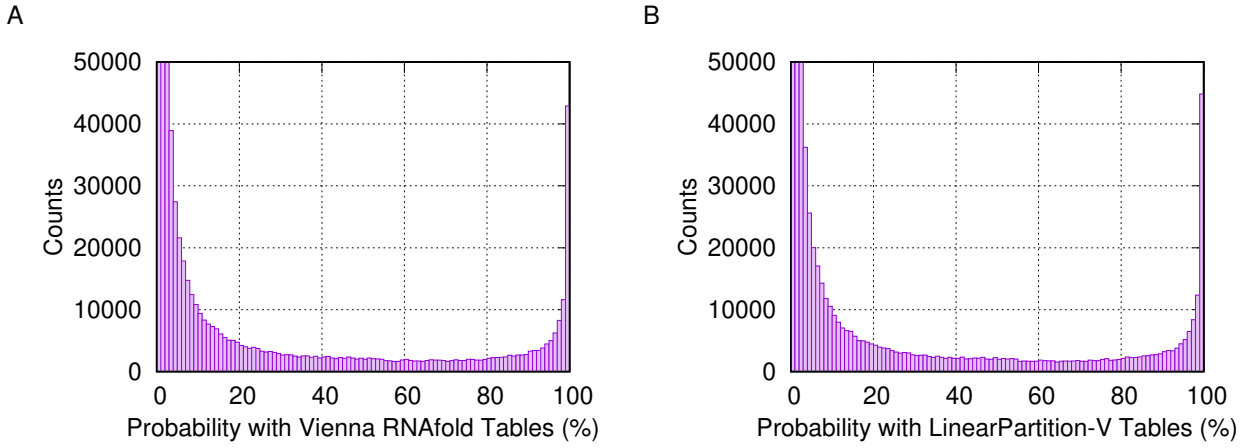


Fig. SI 7. Pair probability distributions of Vienna RNAfold and LinearPartition-V are similar. **A**: pair probability distribution of Vienna RNAfold; **B**: pair probability distribution of LinearPartition-V. The count of LinearPartition-V in bin [99,100) is slightly bigger than Vienna RNAfold, while the count in bin [0,1) (cut here at 50,000) is much less than Vienna RNAfold (2,068,758 for LinearPartition-V and 48,382,357 for Vienna RNAfold).