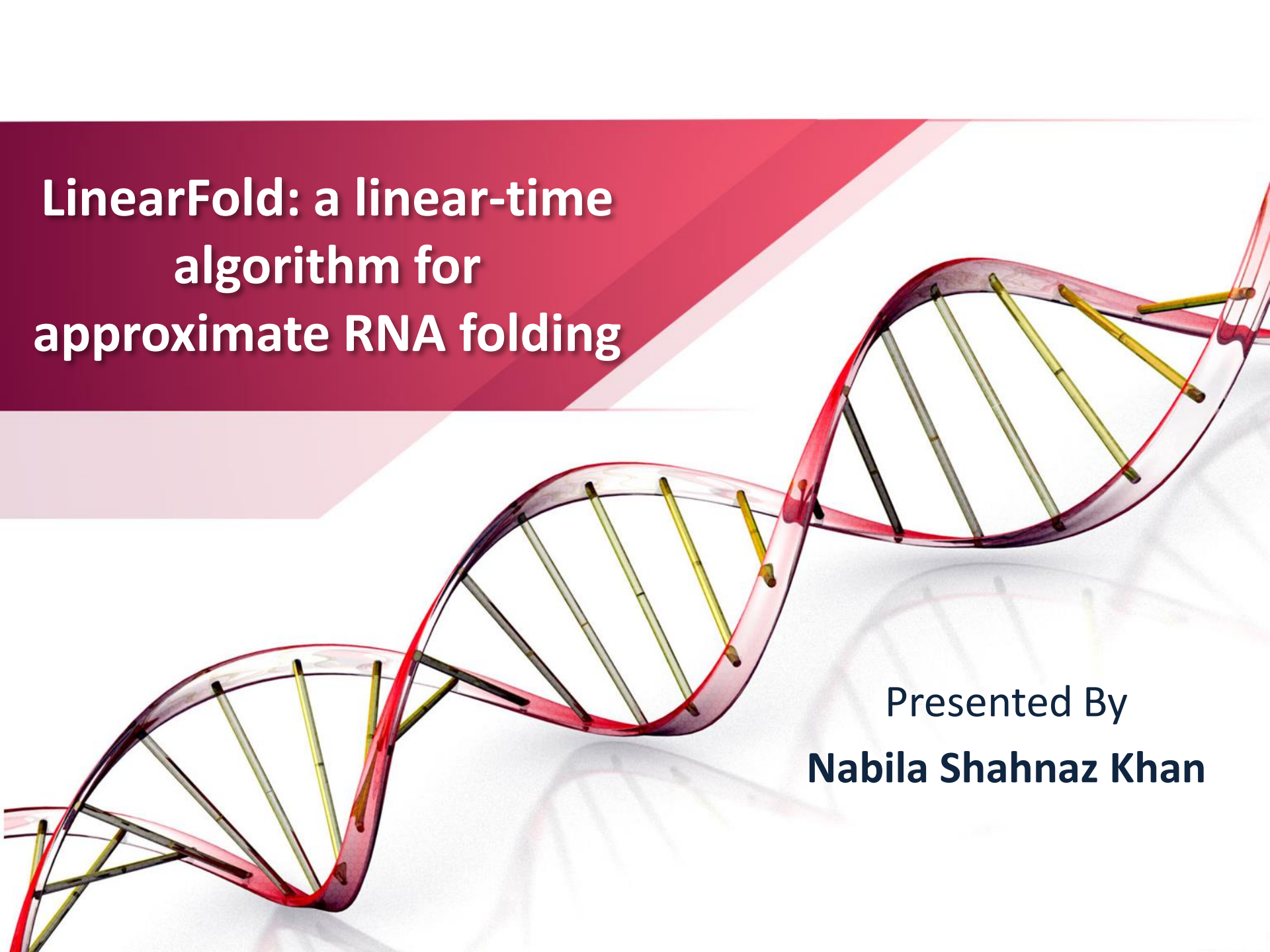
LinearFold: a linear-time algorithm for approximate RNA folding



Presented By

Nabila Shahnaz Khan

Paper to be Presented

- **Paper Title:**

LinearFold: linear-time approximate RNA folding by 5’-to-3’ dynamic programming and beam search

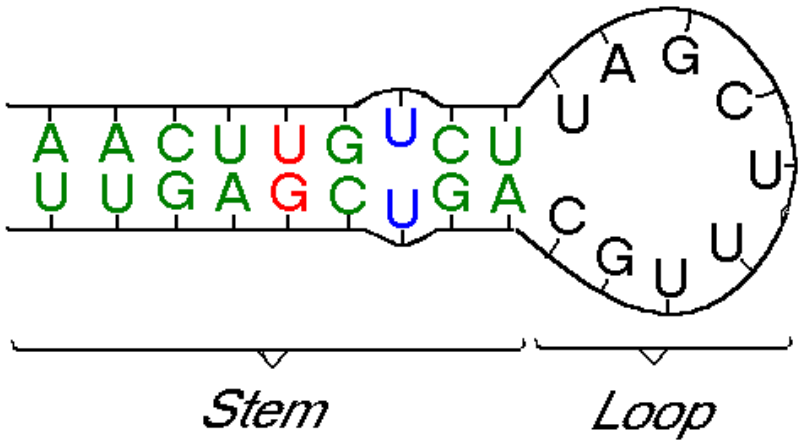
- **Authors:**

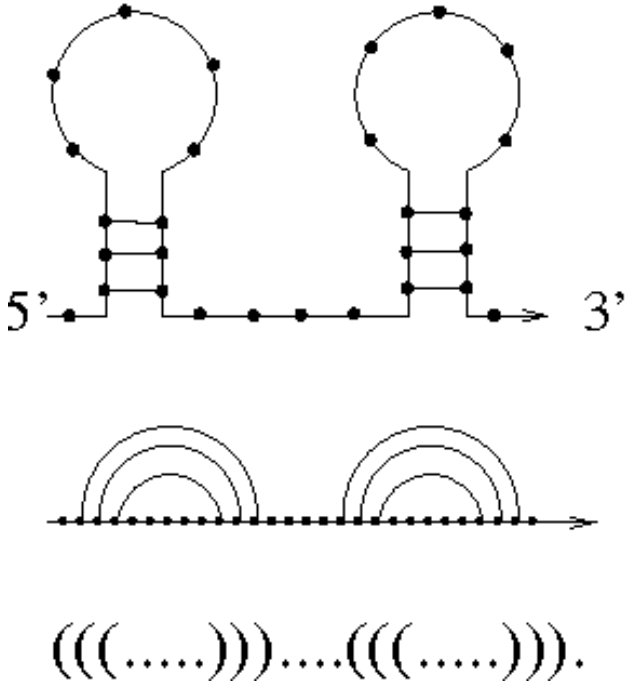
Liang Huang, He Zhang, Dezhong Deng, Kai Zhao, Kaibo Liu, David A. Hendrix, David H. Mathews

- **Published on:** Bioinformatics Journal, July 2019

Brief Idea on RNA Folding

- A linear RNA folds to acquire secondary structure
- In Secondary structure they form stacks (more than one base-pair) and loops (unpaired bases)





Problem Statement

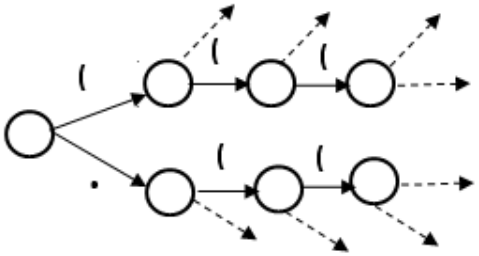
- **Applications of RNA secondary structures:** Predicting RNA function, similarity, evolution, sequence conservation
- **Various techniques for predicting RNA secondary structure** – maximum base-pairs, minimum free energy (Vienna RNAfold), stochastic context-free grammars (CONTRAfold), generalized centroid estimator (CentroidFold); **Runtime O(n3)**
- **Faster linear time algorithms** such as Rfold, Vienna RNAplfold, LocalFold, but they have some constraints
- The paper **proposed a Linear time algorithm O(nblogb)** named **LinearFold** which can predict RNA secondary structures **without imposing any constraints** and also has **high accuracy**

Interesting Facts

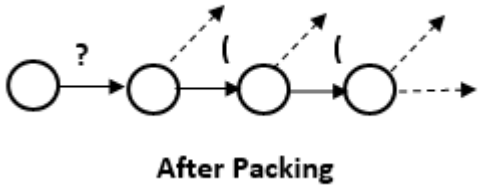
- **Functions** of RNAs are deeply **related** to their **secondary structures**
- **Predicting RNA secondary structure in linear time** using Dynamic Programming concept of well known Zuker's Algorithm **while runtime of Zuker's Algorithm is $O(n^3)$; space complexity** reduced to **$O(n)$** from **$O(n^2)$**
- **Doesn't impose any constraints** (local folding, fixed length of base-pair) on the output to achieve this linear runtime
- Still have a **high accuracy** compared to the state-of-art tools
- Works even better for **longest sequence families** and **sequences with long-range base pairs**!!!

Steps taken to Address the problem

- Used **left-to-right method** instead of traditional bottom-up for dynamic programming
- **Merges states with identical condition** (similar number of unpaired opening brackets till now) to reduce the number of considered states
- **Uses the concept of packing** from computational linguistics to reduce runtime even further



Before Packing



After Packing

- Finally, **uses beam search** to make the runtime linear

Dataset Used

❖ **The ArchiveII dataset:**

- A set of 3847 RNA sequences with known secondary structures
- The set includes small subunit ribosomal RNA (22s), large subunit ribosomal RNA (6s), 5S ribosomal RNA (1283s), group I introns (98s), signal recognition particle RNA (928s), RNase P (454s), tRNA (557s), tmRNA (462s), and telomerase RNA (37s)
- contains sequences of 3000*nt* or less

❖ **A sampled subset of RNAcentral**:

- A database of non-coding RNA (ncRNA) sequences
- currently holds 10.2 million distinct ncRNA sequences
- Has many longer sequences, with the longest being 244296*nt*

Computational Technique Used

- **Uses Dynamic Programming**
- **Implemented in C++**
- **Two versions** of LinearFold Algorithm

i. **LinearFold-V :** uses minimum free energy model (similar to Vienna RNAfold)

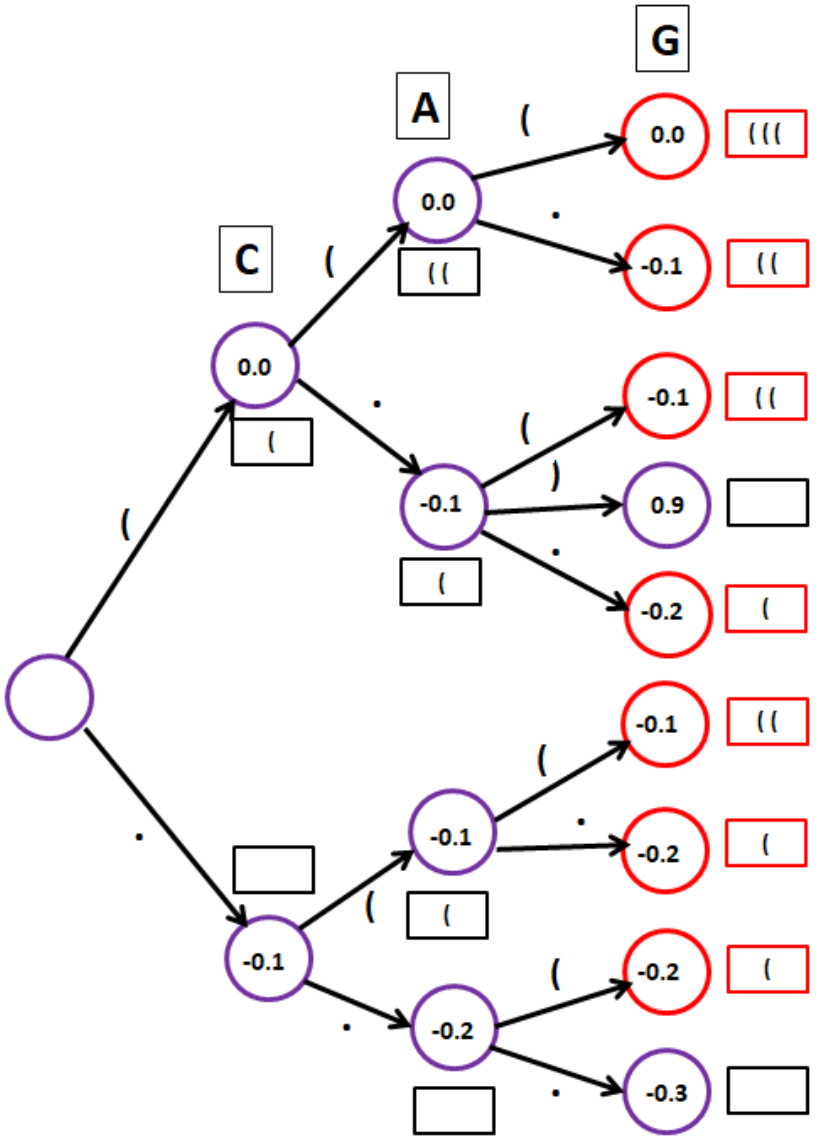
ii. **LinearFold-C:** uses machine learned model (similar to CONTRAfold)

Four stages of Algorithm:

Initially stack empty, score 0

1. Brute force search $O(3^n)$:

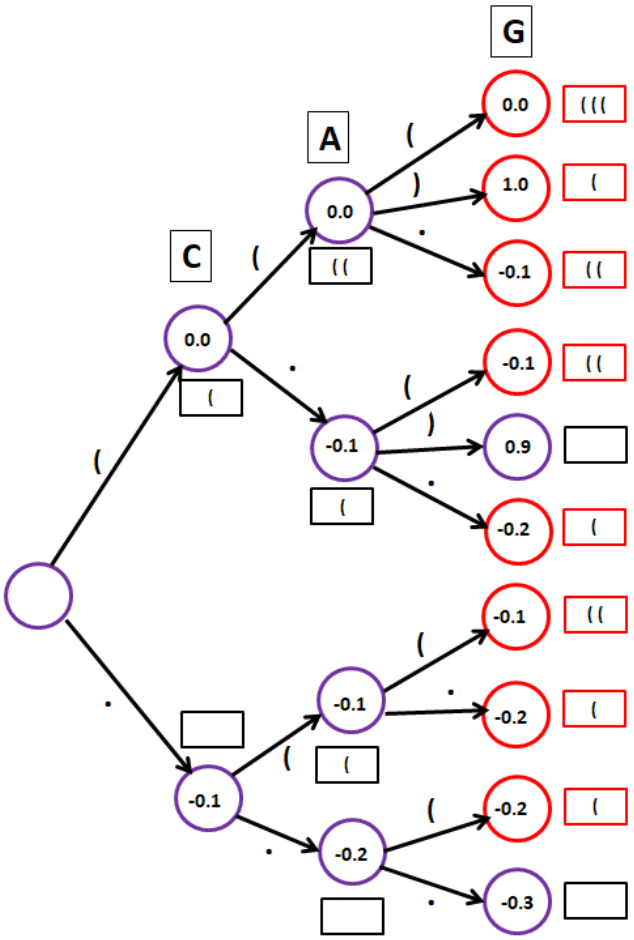
- Scans RNA sequence from **left-to-right**
- Perform one of the three operations- push, pop skip
- If push, then pushes opening bracket inside stack, adds 0 to score
- If skip, add -0.1 to score
- If nucleotide (with top bracket in stack) complementary to current nucleotide add +1 to score, pop '(' from stack

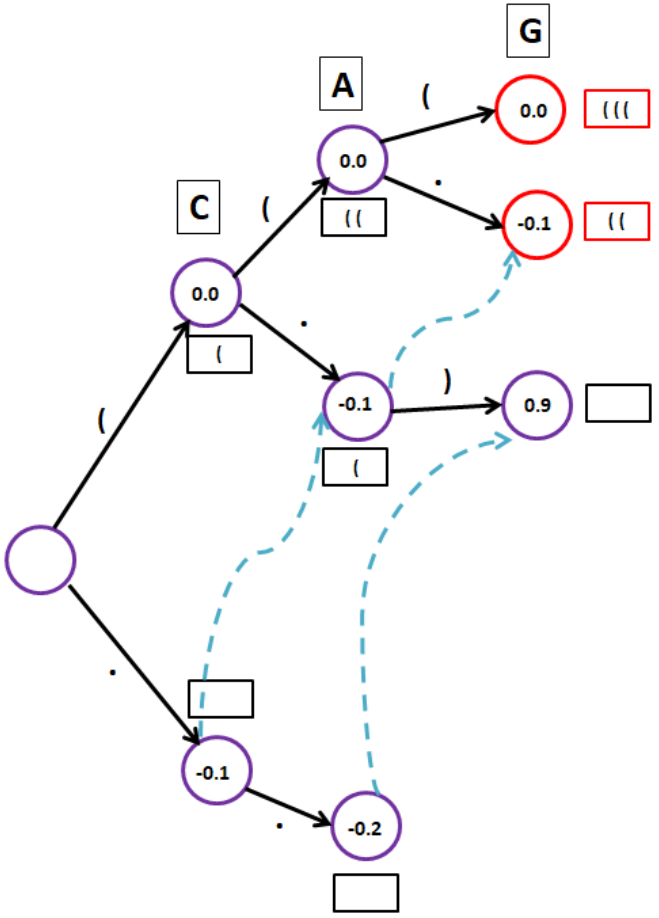


Computational Technique Used

2. Merge Stacks with Identical stacks $O(2^n)$:

- States having same stacks are merged, in this case state with the higher score is kept

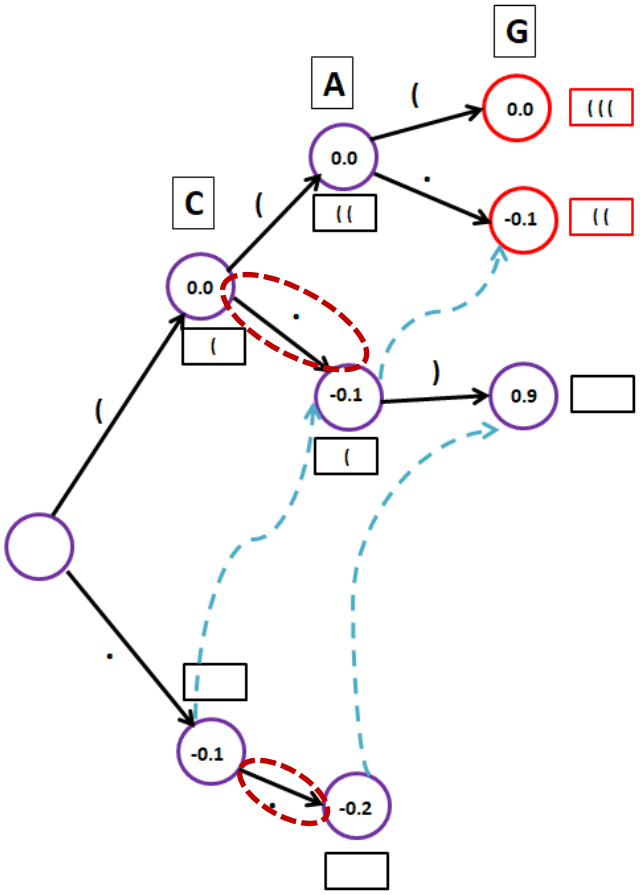


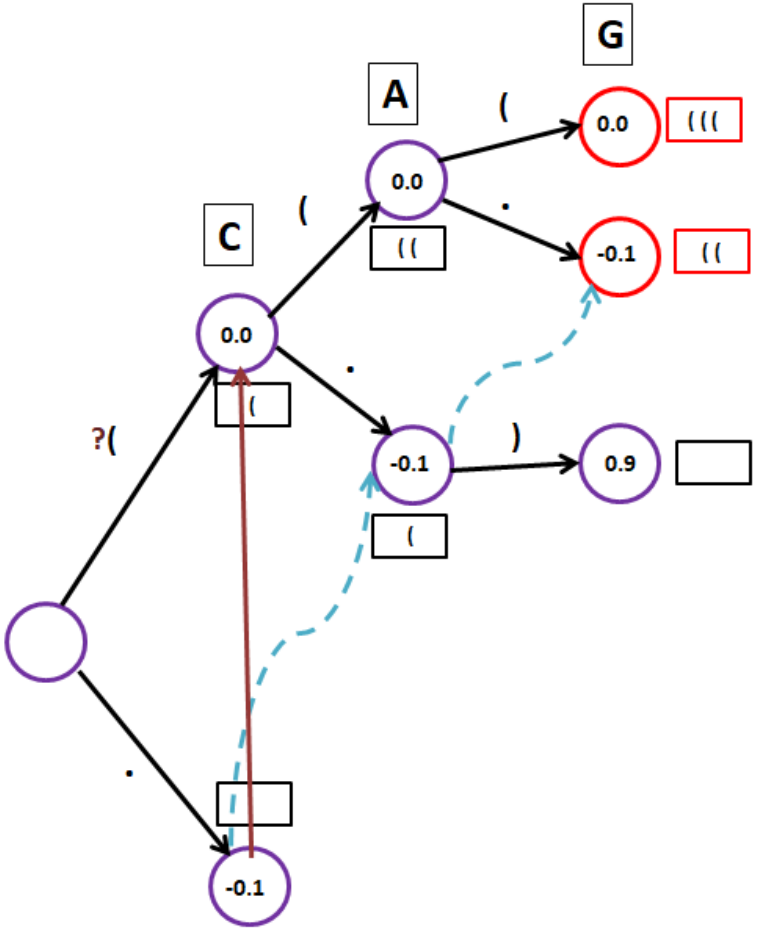


Computational Technique Used

3. Pack Temporarily Equivalent Stacks $O(n^3)$:

- Different states with same stack top can be packed as they are equivalent until top open bracket is closed

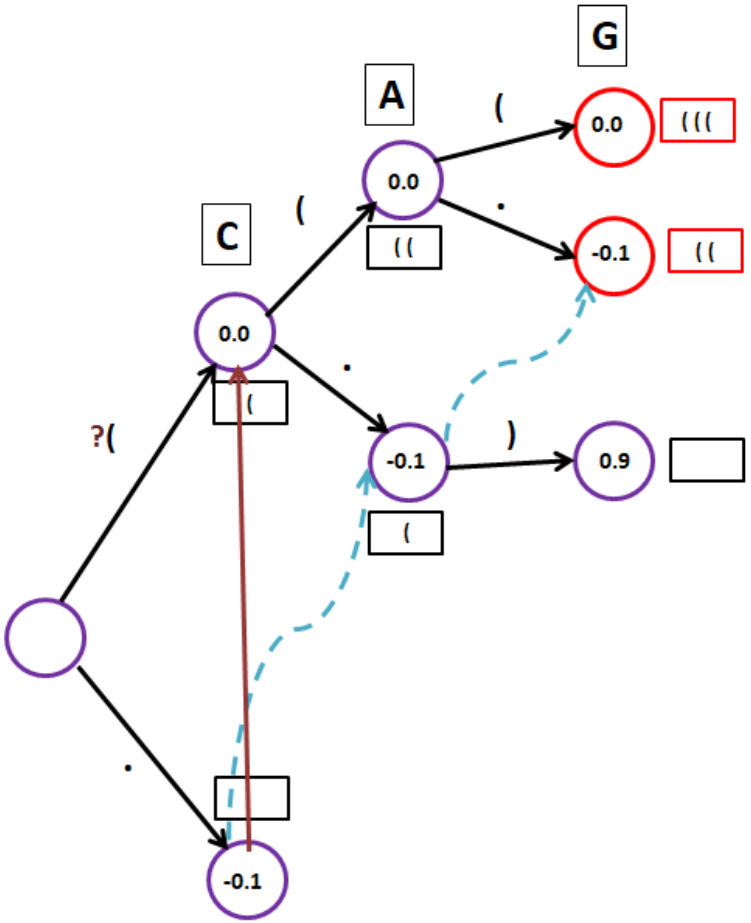


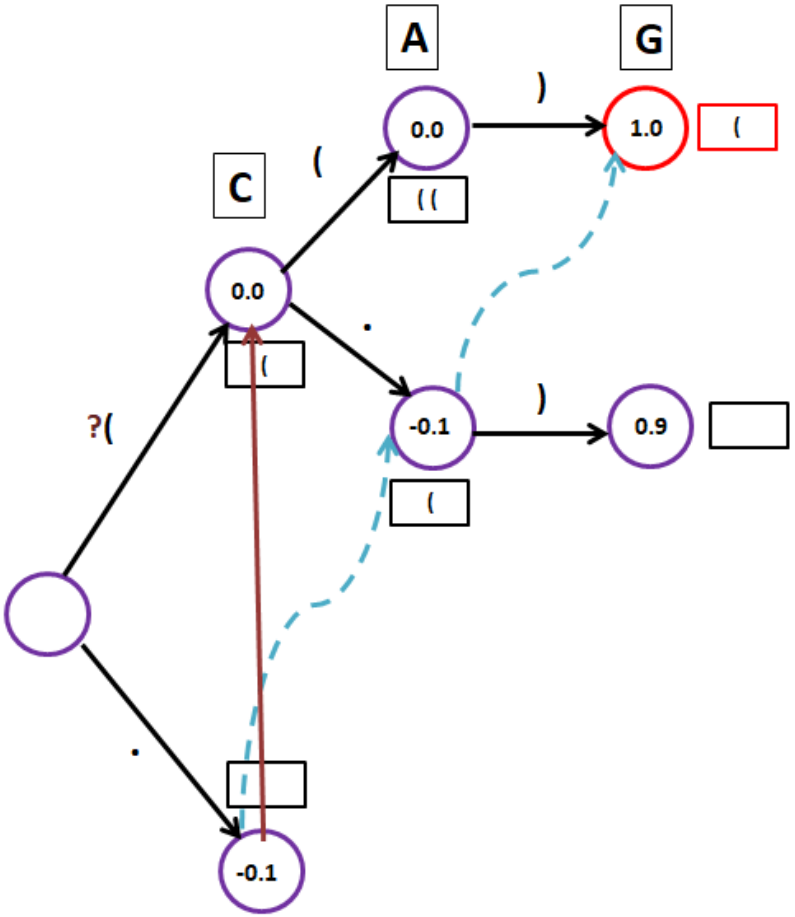


Computational Technique Used

4. Beam Pruning, O(nblogb) ≈ O(n):

- If beam size, b = 2, then only top 2 states will be considered, others will be pruned

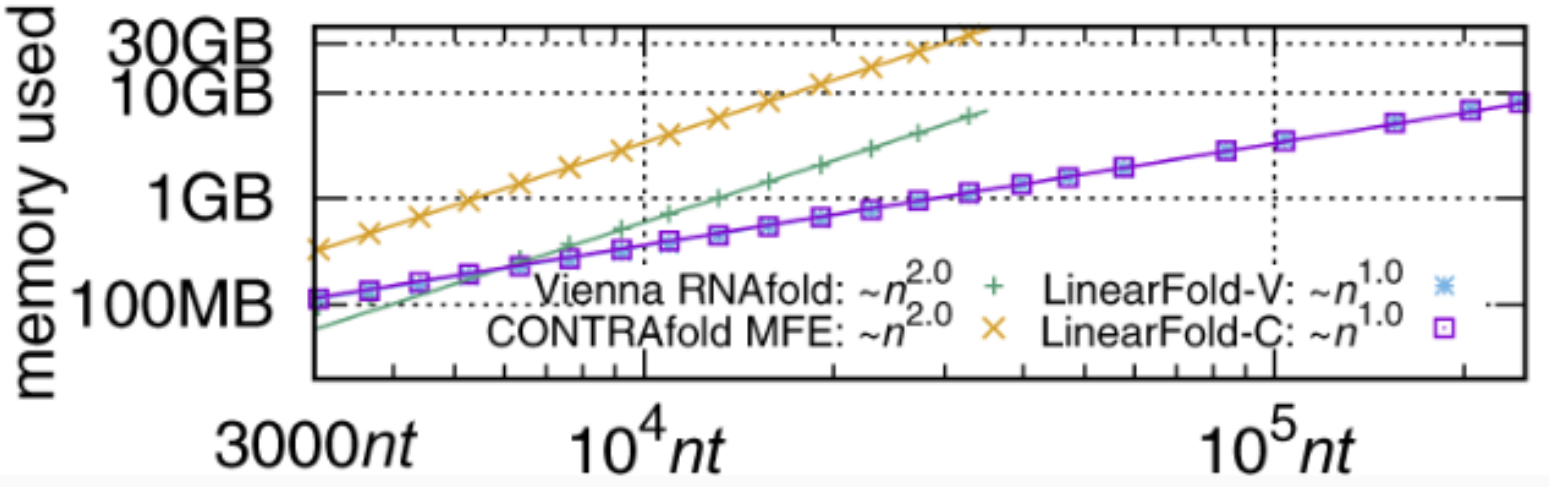




Result Analysis

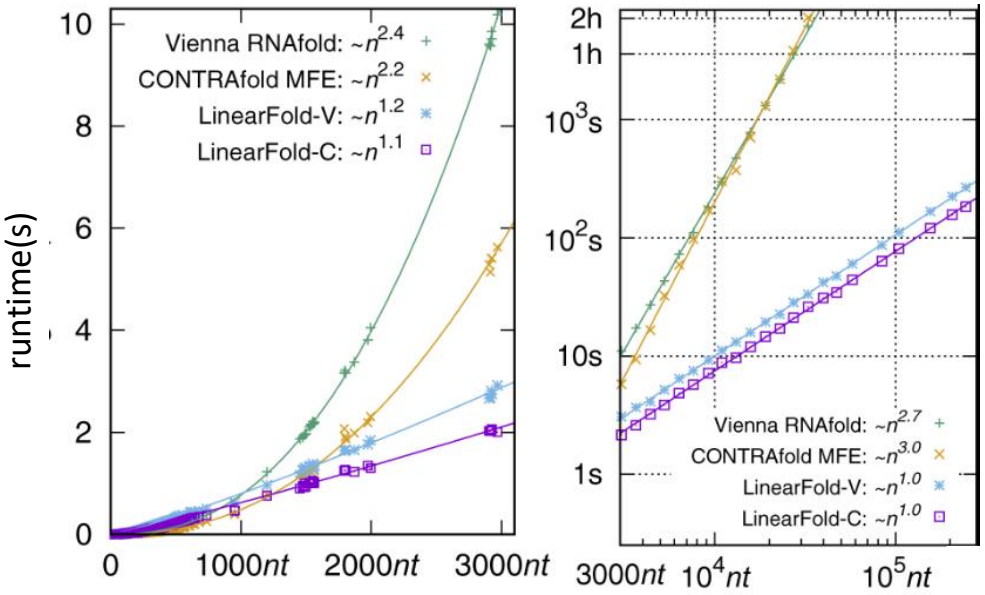
- Compared with two state-of-the-art tools: **CONTRAfold** (Version 2.02) and **Vienna RNAfold** (Version 2.4.10)
- Compared **Space** complexity, **Time** Complexity and **Accuracy** (based on **PPV** (positive predictive value) and **sensitivity**)

❑ Space Complexity Comparison:



Result Analysis

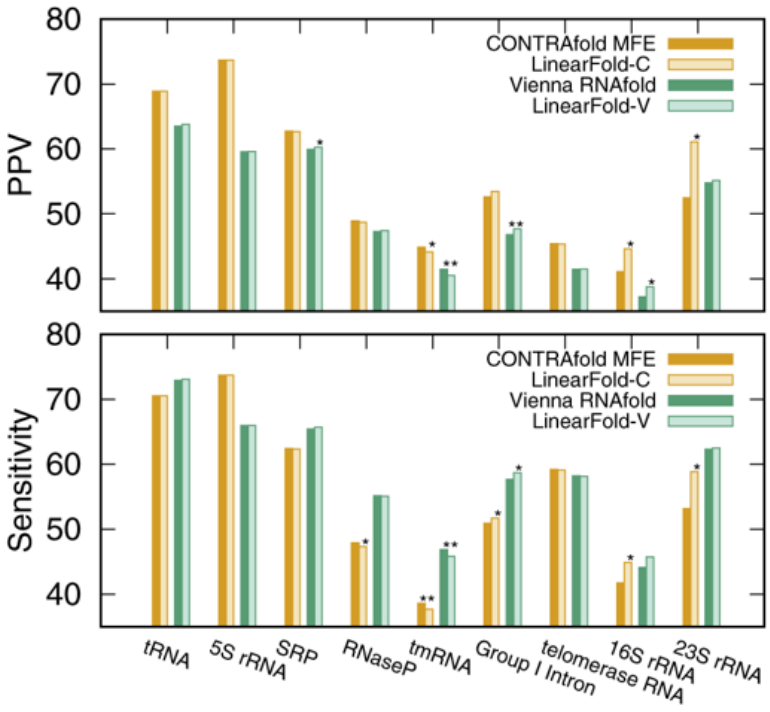
Time Complexity Comparison:



Accuracy Analysis:

$$\textbf{PPV} = \frac{Total\ No.\ of\ Pairs\ (TP)}{No.\ of\ Predicted\ Pairs\ (TP+FP)} \qquad \textbf{Sensitivity} = \frac{No.\ of\ Known\ Predicted\ Pairs}{No.\ of\ Total\ Predicted\ Pairs}$$

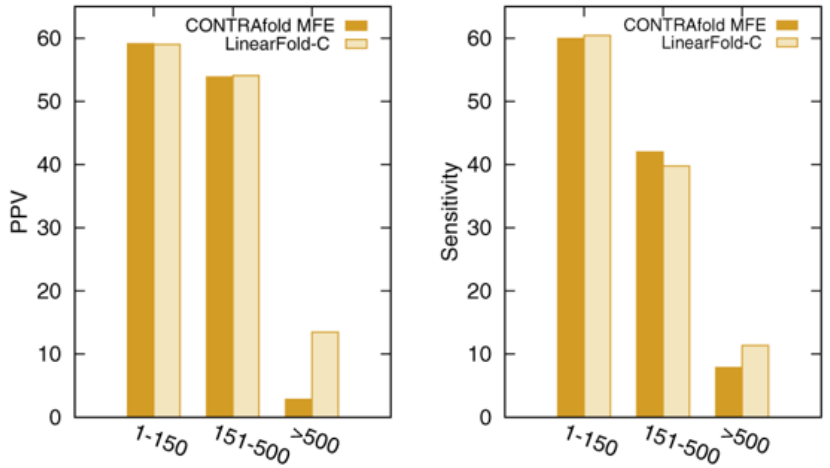
Result Analysis



B

Overall	PPV	Sensitivity
CONTRAfold MFE	54.51	55.36
LinearFold-C	55.84 (+1.3)	56.24 (+0.9)
Vienna RNAfold	50.22	58.74
LinearFold-V	50.51 (+0.3)	58.97 (+0.2)

C



Summary:

- LinearFold-C outperforms CONTRAfold, PPV 1.3% greater and sensitivity +0.9% greater
- LinearFold-V outperforms RNAfold, PPV 0.3% greater and sensitivity 0.2% greater

Why is this Solution Beneficial

- **Runs in linear time O(nblog b)**, so use of longer sequences will become less time consuming
- **Uses linear space**, so will be able to handle larger and longer sequence sets
- In spite of using linear time and memory, **accuracy** is almost as **high** as **CONTRAfold** and **Vienna RNAfold**
- **For longer sequences**, **performs** even **better** compared to those two tools
- Long-range base pairs are harder to predict. Local folding algorithms can not predict structure for longer base pairs. **In case of long-range base pairs** (>500bp), It is **more accurate** than the baselines
- **Search quality** (average search error per nucleotide) **doesn't degrade** with longer sequences
- **Accuracy** is **stable** for a **beam size b** in between 100 to 200, **default value used is 100**

Limitations

- The **implemented code** requires specific compiler and library support; so still not easily accessible by everyone
- Input **sequence length** has to be **less than 100,000 nt**
- **Beam Search assumes** that at any stage, the phases with highest scores are probable solution. It might not always be the case. Example:

	A	U	C	A	U
State1	(	)	.	(	)
Energy	0.0	1	0.9	0.9	1.9
State2	(	(	.	)	)
Energy	0.0	0.0	-0.1	0.9	1.9
j	1	2	3	4	5

- If b =1 & j=2, it will consider State1 as best solution (max energy) and discard State2. But better solution here is State2 (larger size of stacking pair is more stable)

Some Ideas for further Improvement

❖ Improving Beam Search:

- For beam search, at each stage, only the **b** number of states with maximum energy are considered which might compromise optimal solution. To overcome this, I propose to do beam search based on two factors:

  i) maximum score

  ii) structure with longer base pair possibilities

❖ Implementing the concept of LinearFold in case of Alignment

- Current alignment algorithms are $O(n^2)$
- At each stage, this proposed algorithm will perform one of four operations: Insert, delete, replace, match

Any Questions?

Quiz Question

As we know the **runtime** of **LinearFold** algorithm is linear, **O(nblog b)**. What does **'b'** represent and what's the default value for **'b'**?

a) Base pair, 100
b) Beam size, 100
c) Base pair, 150
d) Beam size, 150