LinearFold

Linear-Time RNA Folding

\[
X \quad \text{GCGGGAAUAGCUCAGUUGGUAGACGCACGACCUUGCAGGUCGGGUGCAGAGUCUGUCUUUCGCCCUCA}
\]

\[
y \quad (((((((........)))..(((((......))))))))..(((((......))))))....)
\]

Liang Huang
Baidu Research USA & Oregon State University

Joint work with Dezhong Deng (Oregon State / Baidu) and Kai Zhao (Oregon State / Google) and David Hendrix (Oregon State) and David Mathews (Rochester)

Stanford University School of Medicine, July 2018
A Bit About Myself…

- my main area is computational linguistics (aka natural language processing)
- where I develop faster (linear-time) algorithms to understand/translate languages
- but I also apply these algorithms to computational structural biology…

Ph.D., 2008
Research Scientist, 2009

Assistant Professor, 2015-
Principal Scientist, 2018-
RNA Structure Prediction and Design

CRISPR/Cas9: gene editing

RNA sequence
GCGGGAAUAGCUCAGUGGUAGACGCACUCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUUCCCUCUCA

structure prediction

RNA secondary structure

RNA 3D structure

M. tuberculosis

M. tuberculosis

RNA 3D structure

CRISPR/Cas9: gene editing

structure prediction

RNA secondary structure

RNA 3D structure

M. tuberculosis
allowed pairs: G-C A-U G-U
assume no crossing pairs

example: transfer RNA (tRNA)

challenge: existing structure prediction algorithms are way too slow: $O(n^3)$
solution: borrow linear-time algorithms from natural language parsing

callout: "example: transfer RNA (tRNA)"

challenge: existing structure prediction algorithms are way too slow: $O(n^3)$
solution: borrow linear-time algorithms from natural language parsing

callout: "example: transfer RNA (tRNA)"
Our Linear-Time Prediction is Much Faster…

10,000nt (~HIV)
4min → 7s

244,296nt (longest in RNAcentral)
~200hrs → 120s

with even slightly better prediction accuracy!!
1955 Chomsky: context-free grammars


1964 Cocke - CKY Parsing: $O(n^3)$

1965 Kasami - CKY Parsing: $O(n^3)$

1967 Younger - CKY Parsing: $O(n^3)$

1965 Knuth: LR Parsing: $O(n)$

1970 Joshi: tree-adjoining grammars

1985 CKY-style TAG parsing in $O(n^6)$

1986 Tomita: Generalized LR Parsing

1985 Shieber: non-CF languages

1980s: $O(n^3)$ CKY for RNA structures

1999: TAGs for RNA pseudoknots

2010: Linear-time DP parsing (Huang & Sagae)

2018: LinearFold: $O(n)$ RNA structure prediction
Current Structure Prediction Method: $O(n^3)$

- Dynamic Programming — $O(n^3)$
- bottom-up CKY parsing
- example: maximize # of pairs (A-U G-C G-U)

![Diagram of dynamic programming approach for structure prediction](image)
How to Fold RNAs in Linear-Time?

- idea 0: tag each nucleotide from left to right
- maintain a stack: push "(" , pop ")" , skip "."  
- exhaustive: $O(3^n)$

(Huang and Sagae, 2010)
How to Fold RNAs in Linear-Time?

- idea 1: DP by merging “equivalent states”
- maintain graph-structured stacks
- DP: $O(n^3)$
How to Fold RNAs in Linear-Time?

- **idea 1**: DP by merging “equivalent states”
- maintain graph-structured stacks
- **DP**: $O(n^3)$

(Huang and Sagae, 2010)
How to Fold RNAs in Linear-Time?

• idea 2: approximate search: beam pruning
  • keep only top $b$ states per step
  • DP+beam: $O(n)$

 Each DP state corresponds to exponentially many non-DP states

graph-structured stack (GSS)
(Tomita, 1986)

(Huang and Sagae, 2010)
Another View: Left-to-Right CKY

- many variants of CKY ~ various topological ordering

\[(S, 0, n)\]

- bottom-up
- left-to-right
- right-to-left

all $O(n^3)$, but the incremental ones can apply beam search to run in $O(n)$
Our Linear-Time Prediction is Much Faster…

10,000nt (~HIV)
4min → 7s

244,296nt (longest in RNAcentral)
~200hrs → 120s

with even slightly better prediction accuracy!!
On to details...
An Example Path

0 1 2 3 4 5
C CC CCA CCAG CCAGG

15
Version 1: Exhaustive Search $O(3^n)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>CC</td>
<td>CCA</td>
<td>CCAG</td>
<td>CCAGG</td>
<td></td>
</tr>
</tbody>
</table>
Version 1: Exhaustive Search $O(3^n)$

```
0  1  2  3  4  5
C  CC  CCA  CCAG  CCAGG
```
Version 1: Exhaustive Search $O(3^n)$
Version 1: Exhaustive Search $O(3^n)$
Version 1: Exhaustive Search $O(3^n)$
Version 1: Exhaustive Search $O(3^n)$
Idea 1a: Merge Identical Stacks

Merge states with the same full stack (unpaired openings): “Equivalent States”
Version 2: Merge by Full Stack $O(2^n)$
Version 2: Merge by Full Stack $O(2^n)$

merge states with identical stacks

exhaustive

full-stack merge
Version 2: Merge by Full Stack $O(2^n)$

exhaustive

full-stack merge

+ full stack merge

DP

$O(2^n)$
Idea 1b: Merge “Temporary Equivalents”

Merge states with the same **top of the stack** (last unpaired opening):

“Temporarily Equivalent States”
Version 3: Merge by Stack Top $O(n^3)$

packing temporarily equivalent states
Version 3: Merge by Stack Top $O(n^3)$
Version 3: Merge by Stack Top $O(n^3)$

<table>
<thead>
<tr>
<th>C</th>
<th>CC</th>
<th>CCA</th>
<th>CCAG</th>
<th>CCAGG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ε</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- Stack Top
- Merge by Stack Top
- Complexity: $O(n^3)$
Version 3: Merge by Stack Top $O(n^3)$
Version 3: Merge by Stack Top $O(n^3)$
Close Up Look at Two Paths
Close Up Look at Two Paths

Step 1: full-stack merge

Step 2: temporarily merge states with the same stack-top
Idea 3: Beam Pruning

full-stack merge

stack-top merge

\[ O(2^n) \]

\[ O(n^3) \]
Version 4: DP with Beam Search $O(n)$

stack-top merge

+beam pruning

LinearFold $O(n)$
(approx. DP)
Recap: $O(3^n)$ to $O(n^3)$ to $O(n)$

- 5 search algorithms
  - DP: bottom-up CKY: $O(n^3)$
  - left-to-right (exhaustive): $O(3^n)$
  - DP: merge by full stack: $O(2^n)$
  - DP: merge by stack top: $O(n^3)$
  - approx. DP via beam search: $O(n)$
- this is a simple illustration that we just maximize the number of pairs
- our real systems work with complicated feature templates
Connections to Incremental Parsing

- shared key observation: local ambiguity packing
- pack non-crucial local ambiguities along the way
- unpack (in a reduce action) only when needed

psycholinguistic evidence (eye-tracking experiments):

delayed disambiguation

John and Mary had 2 papers each
John and Mary had 2 papers together

Frazier and Rayner (1990), Frazier (1999)

(Huang and Sagae, 2010)
Experiments
Applying Prediction Models on LinearFold

- models from two most widely-used systems
  - CONTRAfold MFE (machine-learned)
  - Vienna RNAfold (thermodynamic)
- we linearized both systems from $O(n^3)$ to $O(n)$

<table>
<thead>
<tr>
<th>efficiency</th>
<th>systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>space</td>
</tr>
<tr>
<td>baselines</td>
<td>$O(n^3)$   $O(n^2)$</td>
</tr>
<tr>
<td>our work</td>
<td>$O(n)$  $O(n)$</td>
</tr>
</tbody>
</table>
Efficiency & Scalability

- tested on two datasets
  - **Archive II** dataset (labeled, like Penn Treebank)
    - #: 2,889, avglen: 222nt; up to 2,927nt
    - used for both efficiency and accuracy evaluations
  - **RNAcentral** dataset (unlabeled, like Gigaword)
    - over 1.3M of unlabeled sequences; up to 244,296nt
    - only used for efficiency/scalability evaluations
**Accuracy**

- use beam=100 for LinearFold
- Tested on Archive II dataset (on a family-by-family basis)
- significant improvements on 3 longer families
- biggest improvements on the longest families: 16S/23S rRNAs

\[
\text{Sensitivity} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \\
\text{PPV} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}
\]

<table>
<thead>
<tr>
<th>%</th>
<th>CONTRAfold</th>
<th>LinearFold-C</th>
<th>Vienna</th>
<th>LinearFold-V</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPV (precision)</td>
<td>54.8</td>
<td><strong>56.9 (+2.1)</strong></td>
<td>50.7</td>
<td>50.9 (+0.2)</td>
</tr>
<tr>
<td>Sensitivity (recall)</td>
<td>55.7</td>
<td><strong>57.1 (+1.4)</strong></td>
<td>59.3</td>
<td>59.5 (+0.2)</td>
</tr>
</tbody>
</table>
Beam Size and Search Quality

- beam search achieves good search quality starting $b=100$
- slightly under-predicts compared to exact search, but close

![Graph showing beam size and search quality](image)

- Vienna RNAfold
- LinearFold-V
- CONTRAfold MFE
- LinearFold-C

![Graph showing average free energy and average model cost](image)

- Vienna RNAfold
- LinearFold-V
- CONTRAfold MFE
- LinearFold-C

![Graph showing number of pairs predicted](image)

- Vienna RNAfold
- LinearFold-V
- Ground Truth
- CONTRAfold MFE
- LinearFold-C
Beam Size and PPV/Sensitivity

- PPV/Sensitivity trade-off as a function of beam size
- stable PPV/Sensitivity around $b=100-150$
Improvements on Long-Range Pairs

- long-distance pairs are well-known to be hard to predict
- but our beam search systems are even better on those
Incremental Folding

- left-to-right vs. right-to-left folding (cotranscriptional folding?)
### Example Predictions: CONTRAfold

<table>
<thead>
<tr>
<th>CONTRAfold MFE</th>
<th>Group I Intron C. Mirabilis (526nt, 143 pairs)</th>
<th>16S rRNA A. Pyrophilus (1564nt, 468 pairs)</th>
<th>23S rRNA E. coli (2904nt, 830 pairs)</th>
</tr>
</thead>
</table>
### Example Predictions: Vienna RNAfold

<table>
<thead>
<tr>
<th></th>
<th>Group I Intron <em>C. Mirabilis</em> (526nt, 143 pairs)</th>
<th>16S rRNA <em>A. Pyrophilus</em> (1564nt, 468 pairs)</th>
<th>23S rRNA <em>E. coli</em> (2904nt, 830 pairs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vienna RNAfold</strong></td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
<tr>
<td><strong>LinearFold V=0</strong></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Web Demo 1 (Beam Sizes)

http://web.engr.oregonstate.edu/~liukaib/demo_json+canvas.html
Web Demo 2 (Live Demo)

http://linearfold.eecs.oregonstate.edu
Bonus Track: Deep Learning for RNA Folding
Existing Models

• Physics-based energy model
  • hundreds of carefully designed and experimentally-measured thermodynamic parameters

• Machine Learning based model
  • claimed “without physics”, but still using physics-based feature templates
  • just learn the feature weights (parameters) from data

• Can we really do it “without physics”?
Deep Learning

- Deep Learning for RNA structure prediction
  - RNNs automatically extract features and learn their weights
  - Completely without physics
  - First deep learning RNA secondary structure prediction algorithm
- We borrow techniques from natural language parsing
  - where deep learning has been very successful since ~2016
  - our group’s “span-based” parsing framework is now dominant
Our Approach: from NLP Span Parsing

- Span differences are taken from an encoder (in this case: a bi-LSTM)
- A span is scored and labeled by a feed-forward network
- The score of a tree is the sum of all the labeled span scores

\[ s_{\text{tree}}(t) = \sum_{(i,j,X) \in t} s(i,j,X) \]

Cross + Huang 2016  \hspace{1cm} Stern et al. 2017  \hspace{1cm} Wang + Chang 2016
Our Approach: RNA structure predict.

- Replace words with nucleotides (A,C,G,U) as inputs
- 3 labels for a span:
  - Left-unpair
    - ( )
  - Left-pair
    - ( )
  - Pair
    - ( )
Experiments

- database: S-Full (Andronescu et al. 2008; 2010)
- preprocessing (cap at 700nt; removing pseudoknots and non-canonical base-pairs); 3245 distinct structures; 80% / 20% split for training / set
- randomly split into training set (2586) and testing set (659)
- can be found https://www.cs.bgu.ac.il/~negevcb/contextfold/

<table>
<thead>
<tr>
<th>system</th>
<th>precision</th>
<th>recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physics-based</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vienna RNAfold</td>
<td>53.35</td>
<td>61.68</td>
<td>57.21</td>
</tr>
<tr>
<td>CONTRAfold (off the shelf)</td>
<td>55.75</td>
<td>53.63</td>
<td>55.75</td>
</tr>
<tr>
<td>Machine learning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTRAfold (retrained)</td>
<td>61.13</td>
<td>65.80</td>
<td>63.38</td>
</tr>
<tr>
<td>Deep learning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our model</td>
<td>77.06</td>
<td>59.55</td>
<td>67.18</td>
</tr>
</tbody>
</table>
Thank you very much!

非常 感谢！
fēi cháng  gǎn xiè

Just google “LinearFold”

eternacon