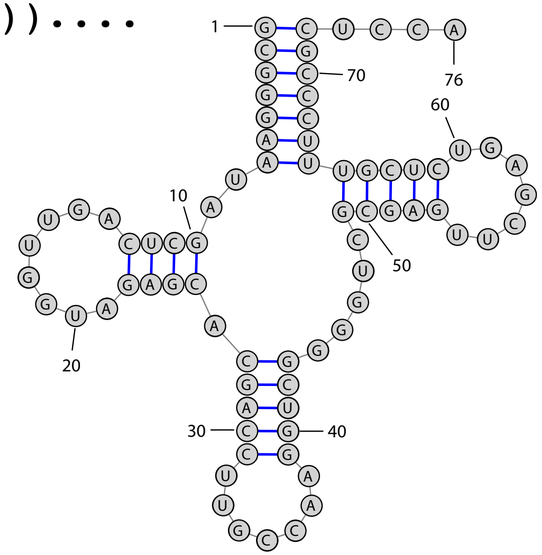
LinearFold: Linear-Time Approximate RNA Folding by 5'-to-3' dynamic programming and beam search

x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

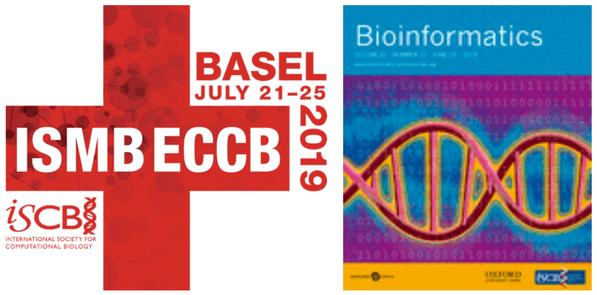
y ((((((((..((((........))))).((((.......)))).....(((((.......)))))))))))....



Liang Huang *

Oregon State University & Baidu Research USA

Joint work with He Zhang **, Dezhong Deng **, Kai Zhao, Kaibo Liu, David Hendrix and David Mathews



ISMB 2019 Proceedings Talk

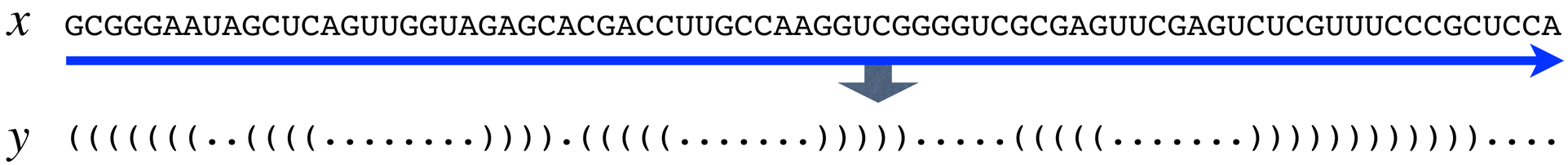
Oregon State University

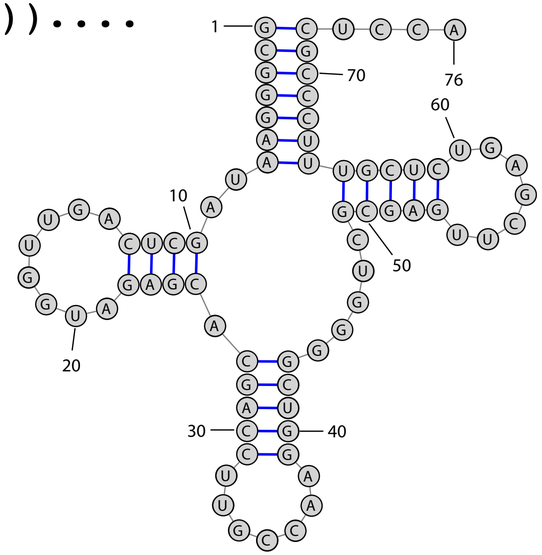
Baidu Research

University of Rochester

* *corresponding author* ** *equal contribution*

LinearFold: Linear-Time Approximate RNA Folding by 5'-to-3' dynamic programming and beam search



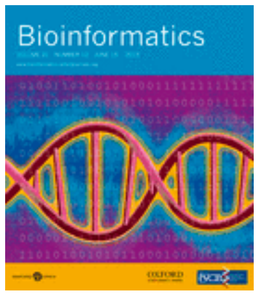


Liang Huang *

Oregon State University & Baidu Research USA

Joint work with He Zhang **, Dezhong Deng **, Kai Zhao, Kaibo Liu, David Hendrix and David Mathews

ISMB ECCB 2019 BASEL JULY 21-25 ISCB INTERNATIONAL SOCIETY FOR COMPUTATIONAL BIOLOGY



ISMB 2019 Proceedings Talk

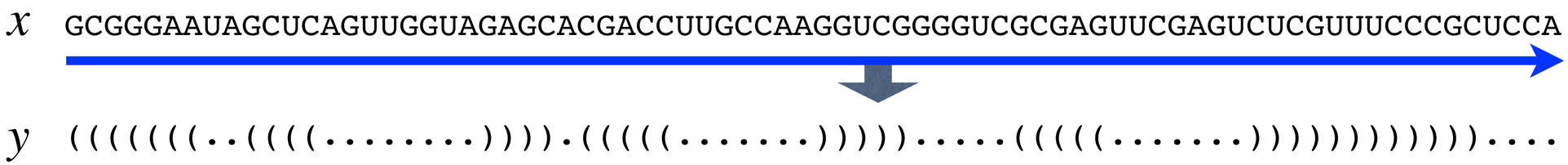
Oregon State University

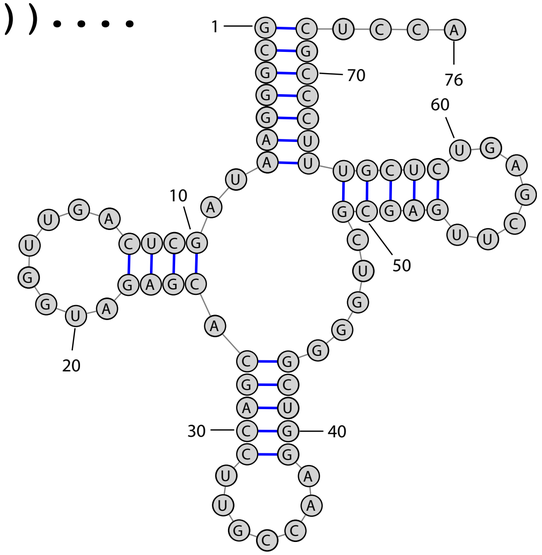
Baidu Research

UNIVERSITY of ROCHESTER

* *corresponding author* ** *equal contribution*

LinearFold: Linear-Time Approximate RNA Folding by 5'-to-3' dynamic programming and beam search



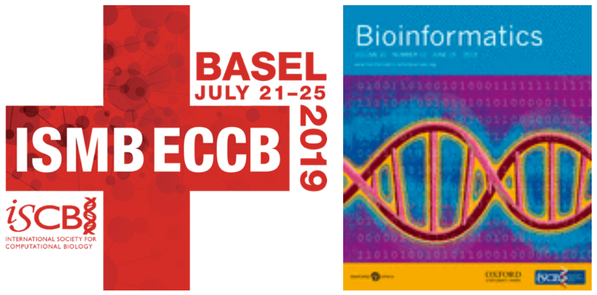


Liang Huang *

Oregon State University & Baidu Research USA

Joint work with He Zhang **, Dezhong Deng **, Kai Zhao, Kaibo Liu, David Hendrix and David Mathews

first $O(n)$ (approx.) RNA folding algorithm & server (linearfold.org) with even higher accuracy than $O(n^3)$ algorithms



ISMB 2019 Proceedings Talk

Oregon State University

Baidu Research

UNIVERSITY of ROCHESTER

* *corresponding author* ** *equal contribution*

RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs
(no pseudoknots)

example: transfer RNA (tRNA)

input x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs
(no pseudoknots)

example: transfer RNA (tRNA)

input x `GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA`

output y `((((((((..((((........))))).((((.......))))).....((((.......))))))))))....`

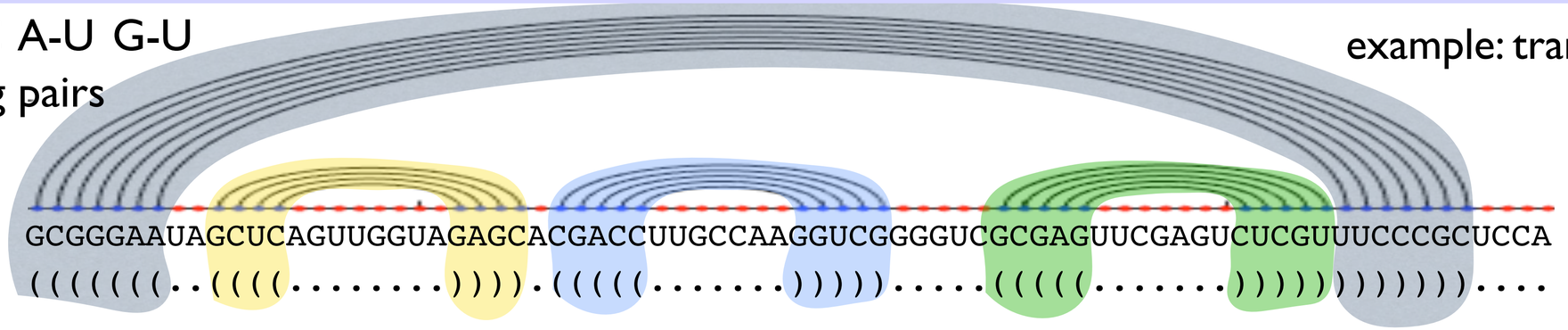
RNA Secondary Structure Prediction

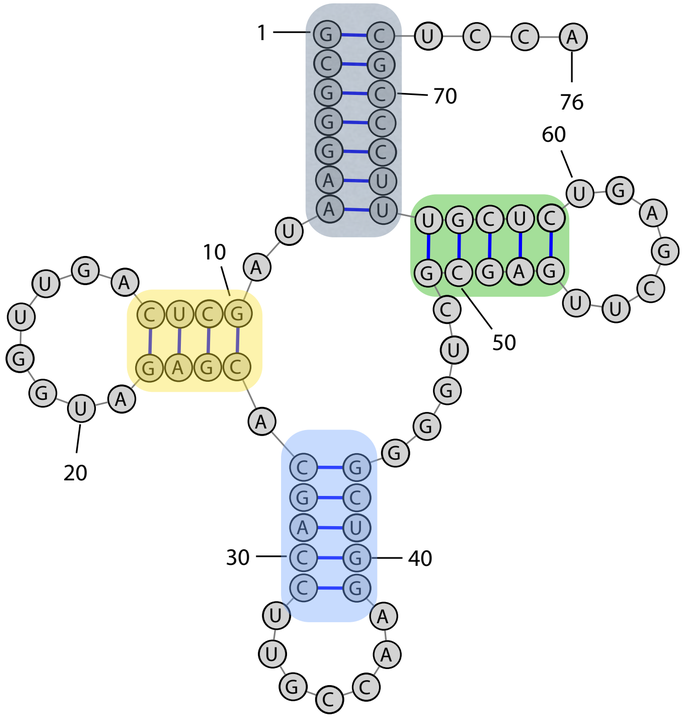
allowed pairs: G-C A-U G-U

assume no crossing pairs

(no pseudoknots)

example: transfer RNA (tRNA)





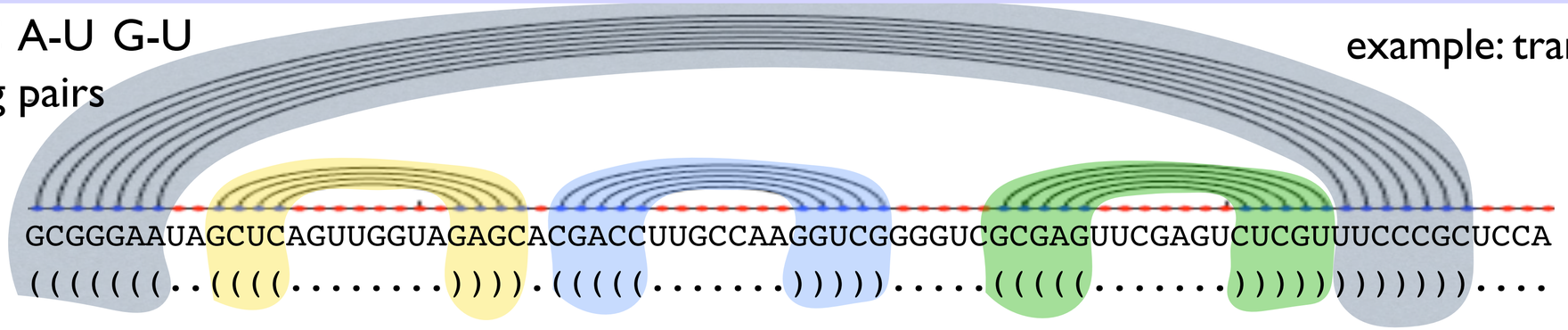
RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs

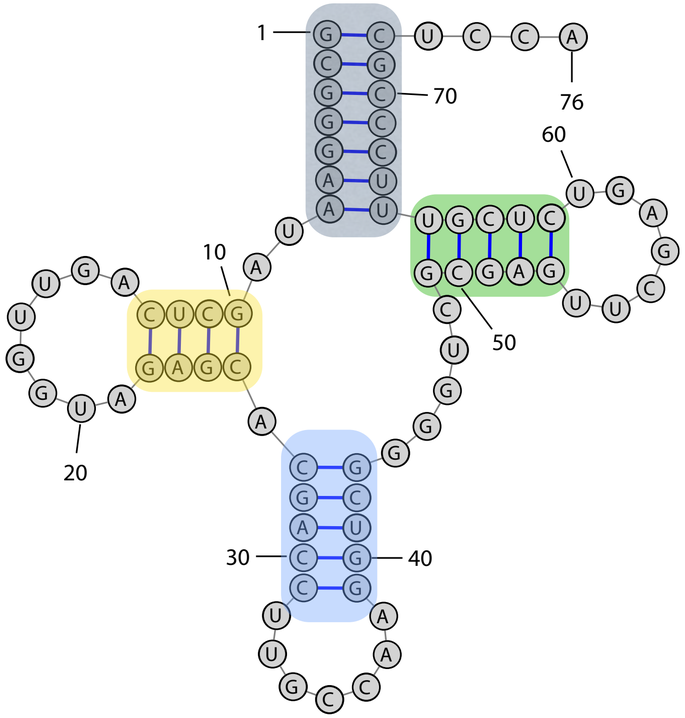
(no pseudoknots)

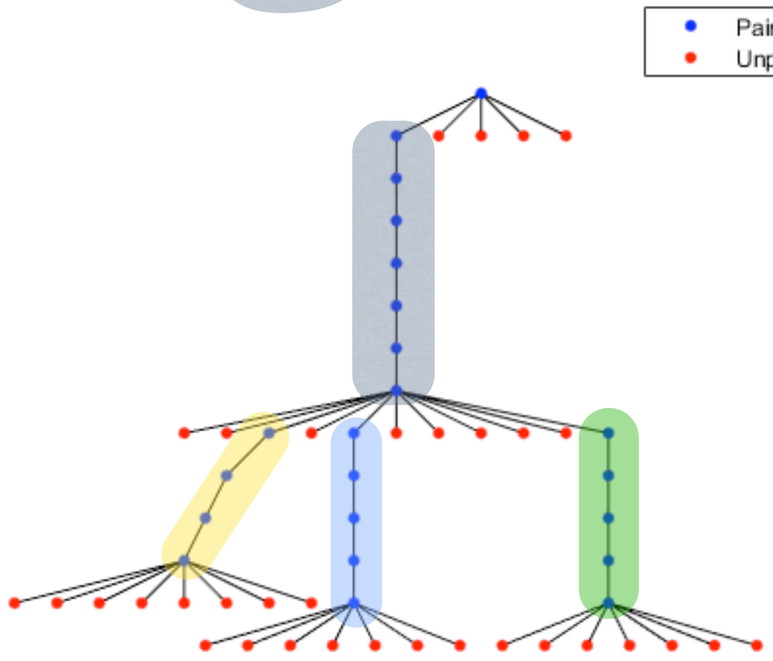
example: transfer RNA (tRNA)



input x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

output y ((((((..((((........))))..((((........))))).....((((.......))))))))).....





parse tree

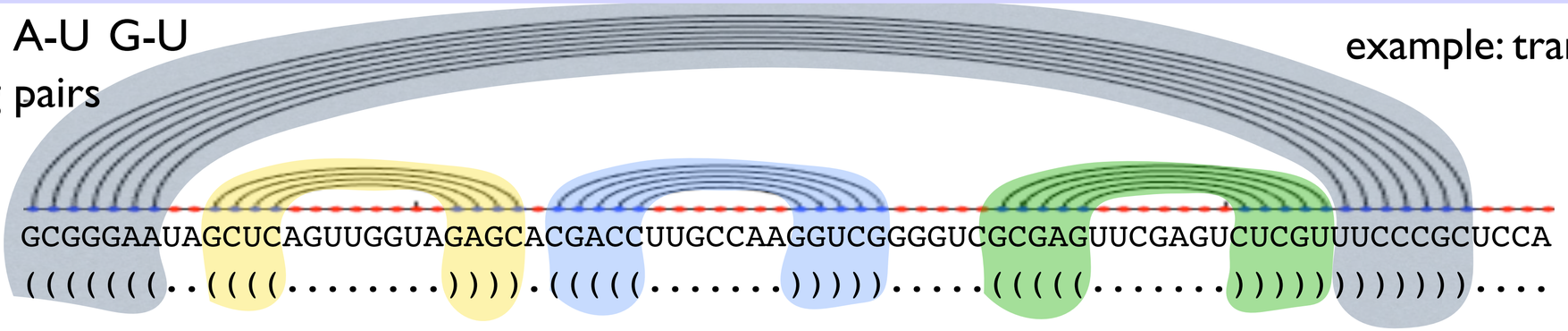
RNA Secondary Structure Prediction

allowed pairs: G-C A-U G-U

assume no crossing pairs

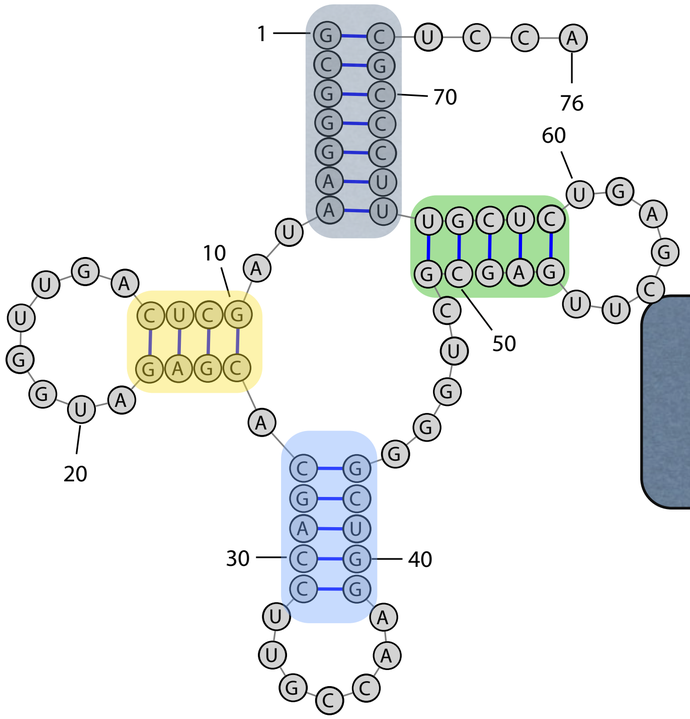
(no pseudoknots)

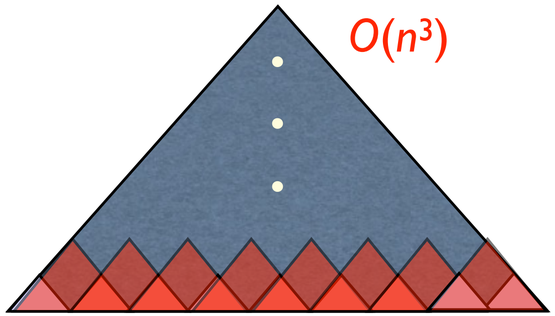
example: transfer RNA (tRNA)



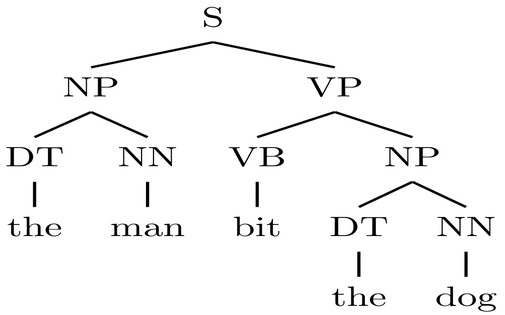
input x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

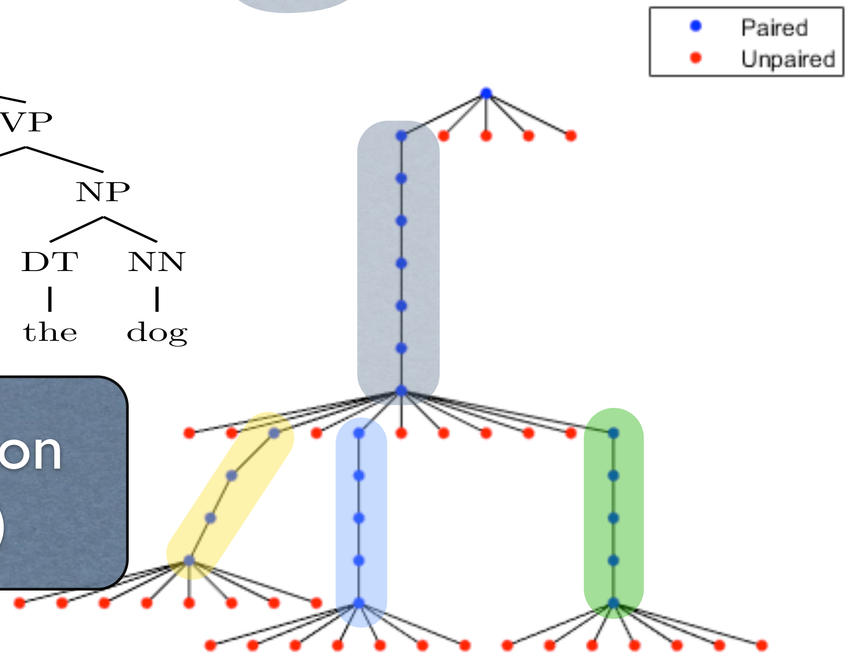
output y ((((((..((((........))))..((((........))))).....((((.......))))))))))....





problem: standard structure prediction algorithms are way too slow: $O(n^3)$





parse tree

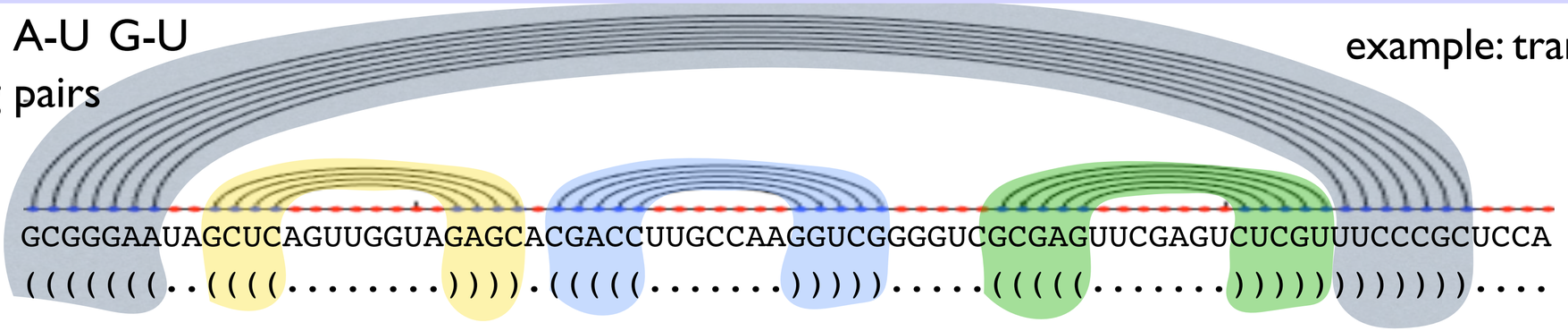
RNA Secondary Structure Prediction

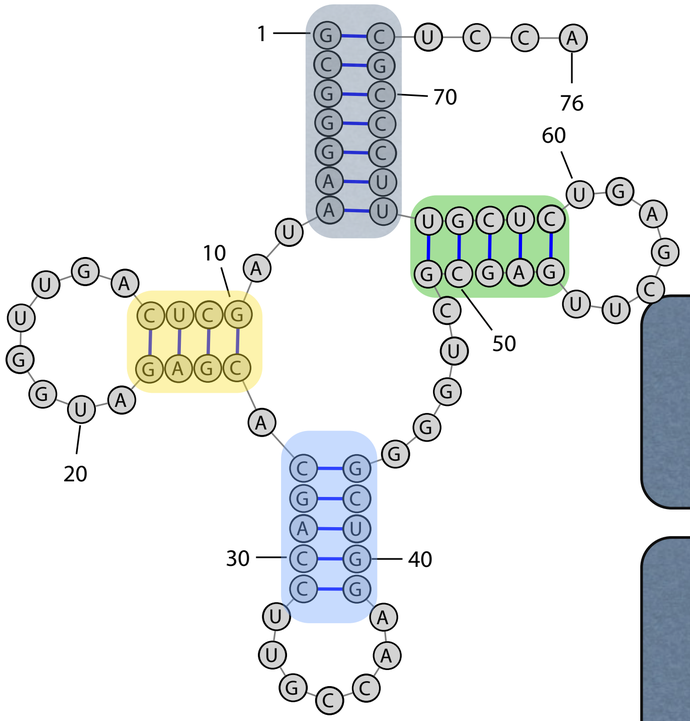
allowed pairs: G-C A-U G-U

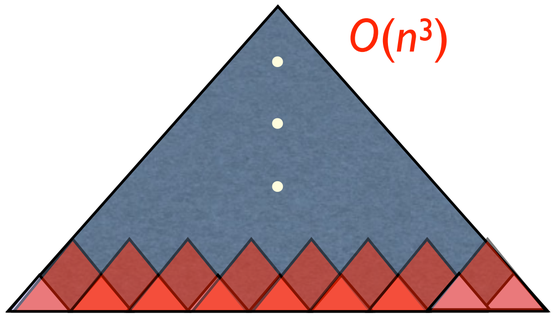
assume no crossing pairs

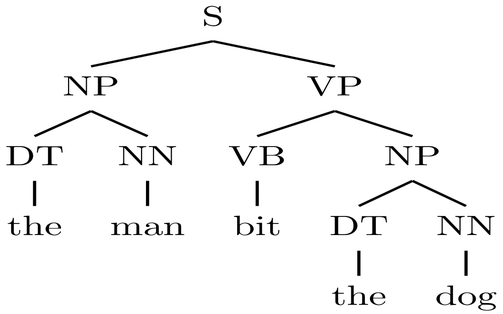
(no pseudoknots)

example: transfer RNA (tRNA)



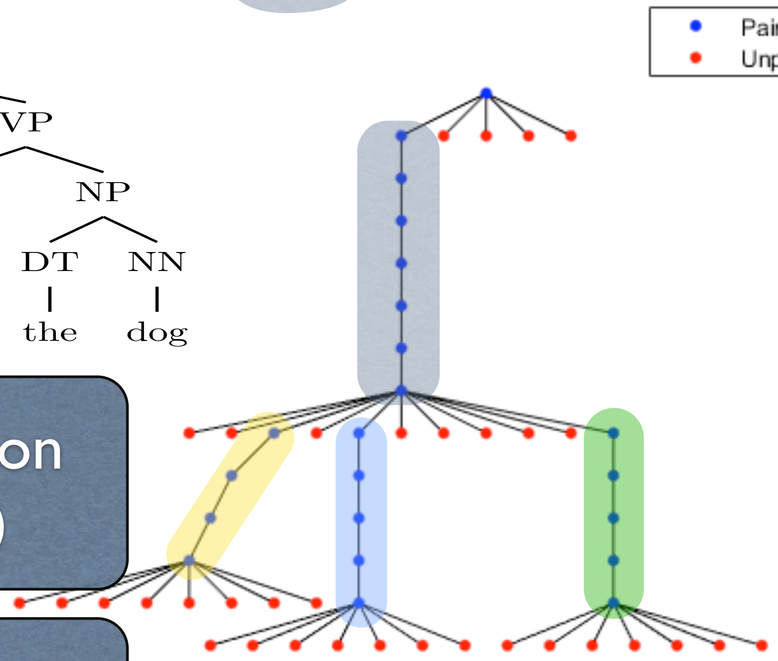






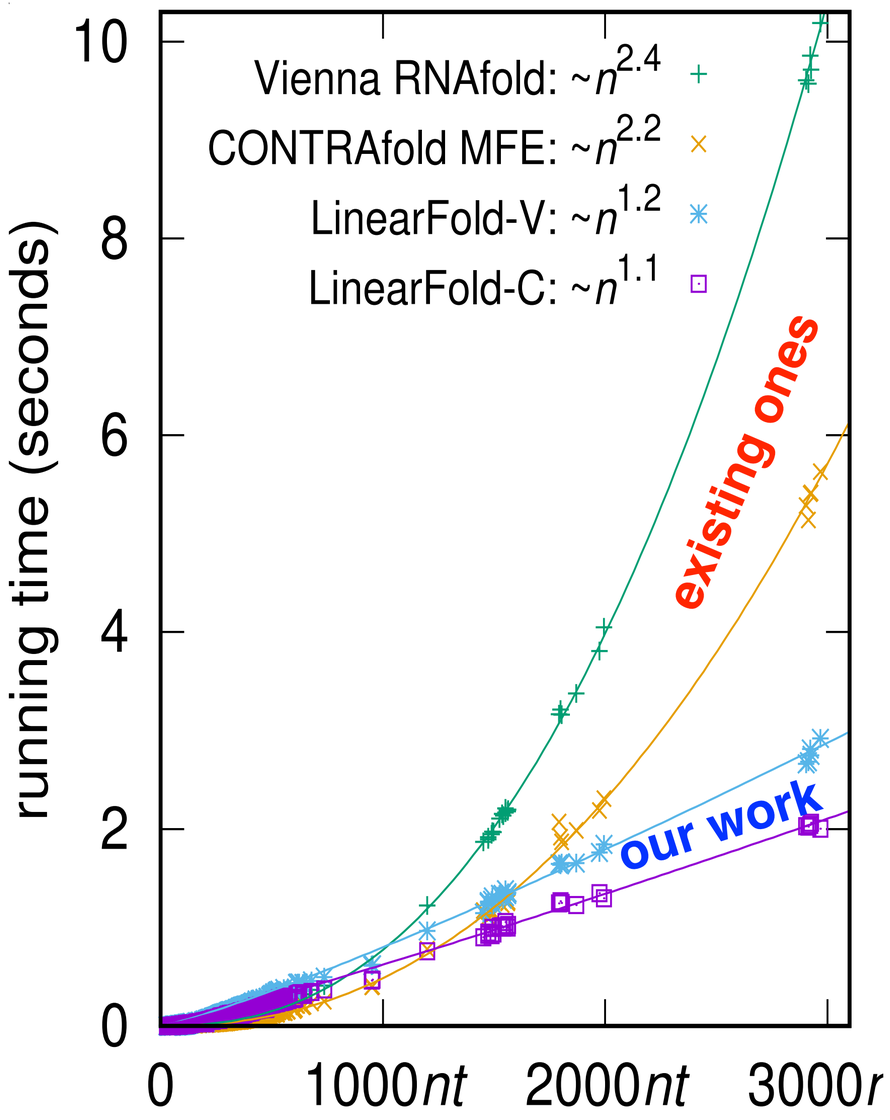
problem: standard structure prediction algorithms are way too slow: $O(n^3)$

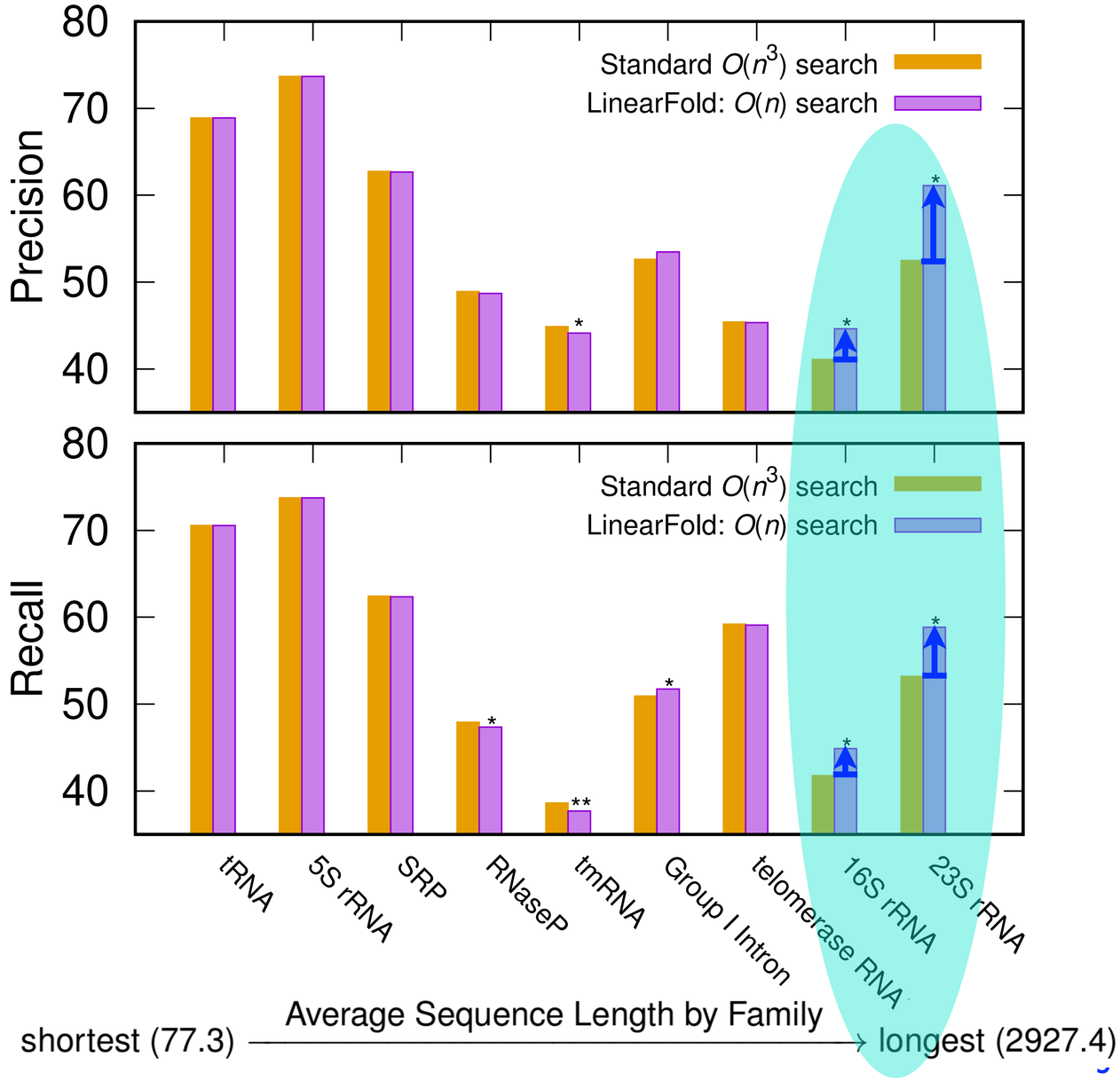
solution: adapt my linear-time dynamic programming algorithms from parsing



parse tree

Results: LinearFold is Much Faster and More Accurate



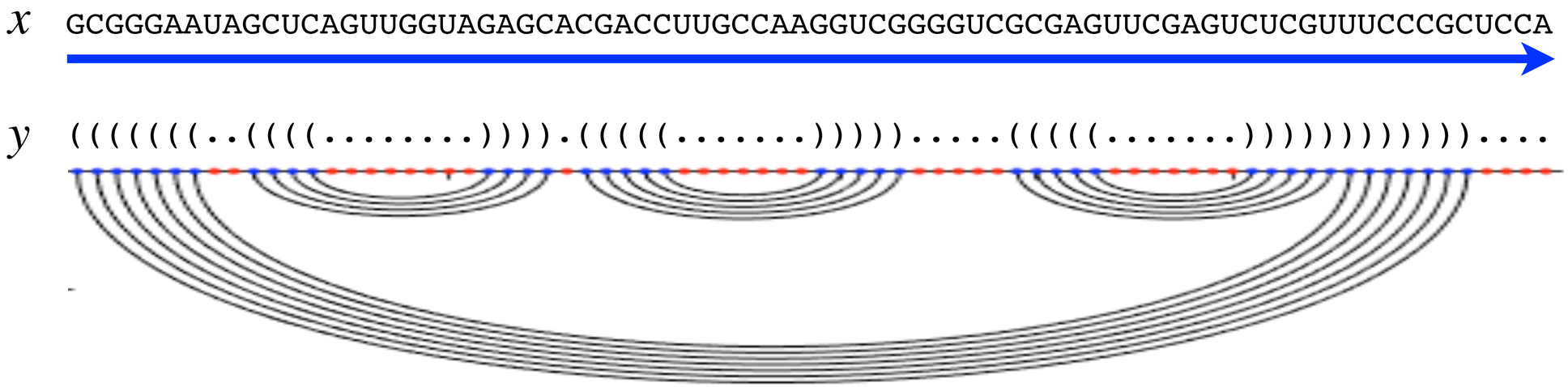


From Linguistics to Biology

x GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

y ((((((..((((........))))).((((.......)))).....((((.......))))))))))....

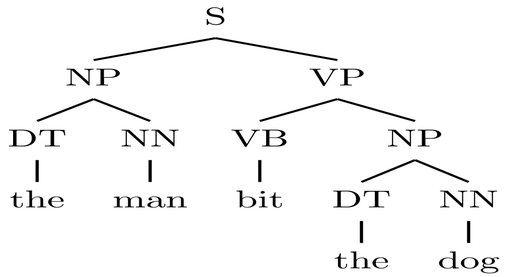
From Linguistics to Biology



Computational Linguistics => Computational Biology

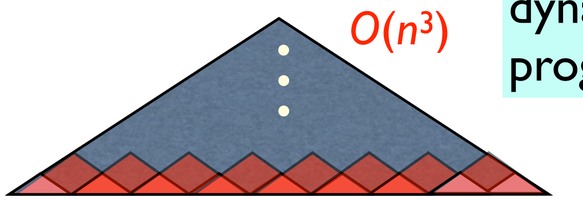
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)

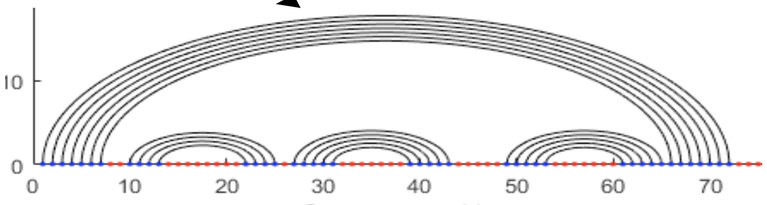


1958 Backus & Naur: CFGs for program, lang.

1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** $O(n^3)$
1967 **Y**ounger / for all CFGs



dynamic programming

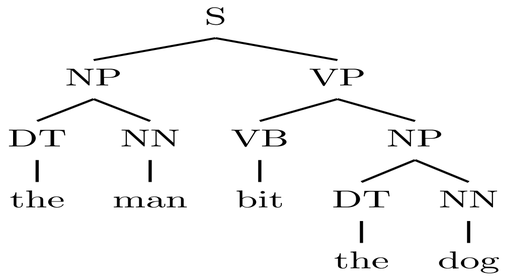


1978: Nussinov $O(n^3)$ RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

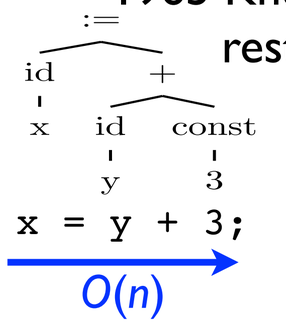
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)

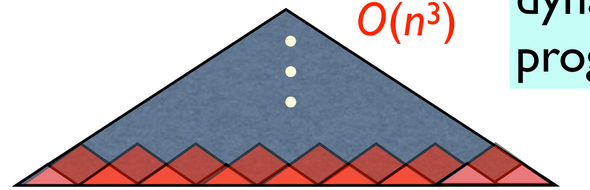


1958 Backus & Naur: CFGs for program, lang.

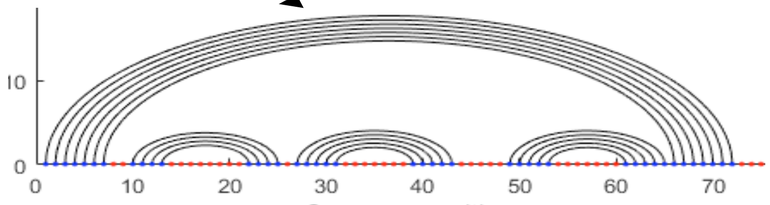
1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***



1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs



dynamic programming

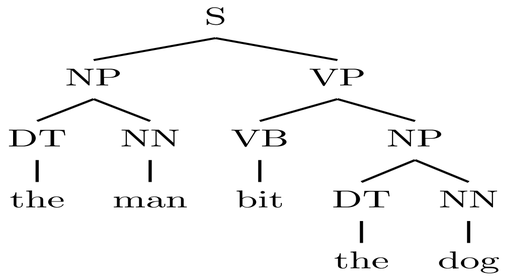


1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

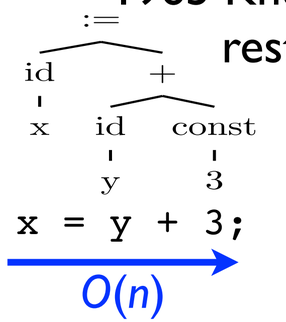
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

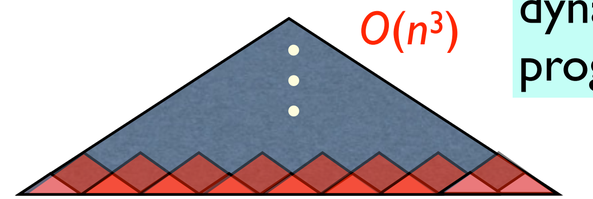
1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***



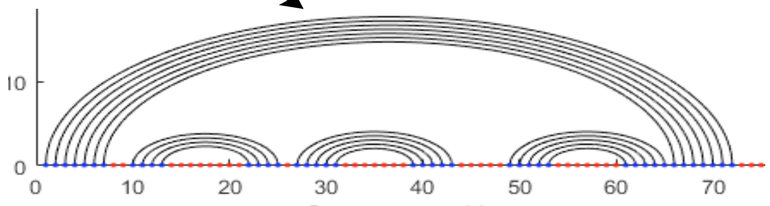
O(n³)

1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs

1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***



dynamic programming

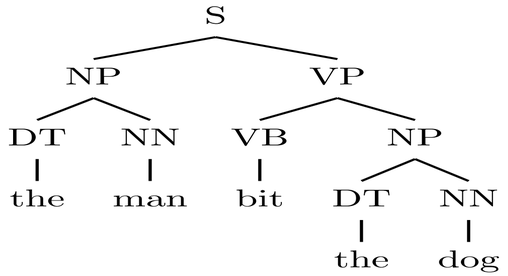


1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

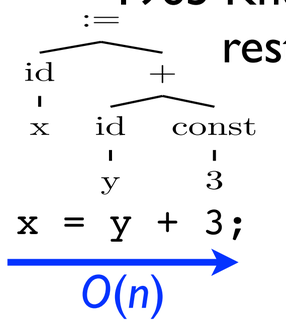
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

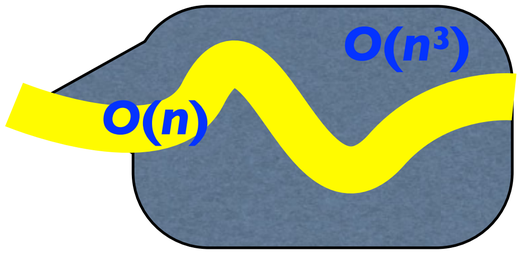
1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***

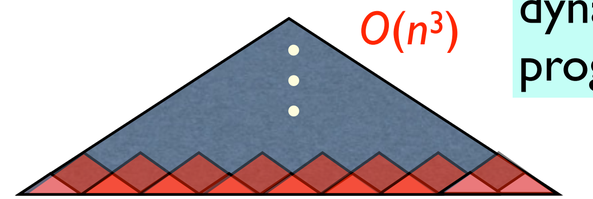




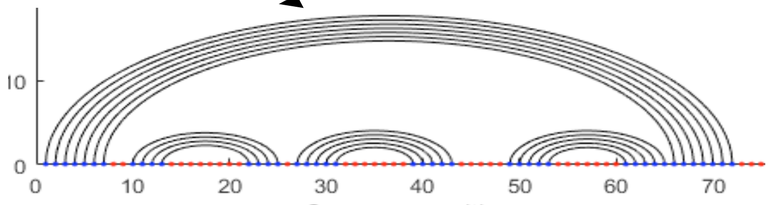
1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs

1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***

2010: Huang & Sagae: ***O(n)*** (approx.) DP for all CFGs



dynamic programming

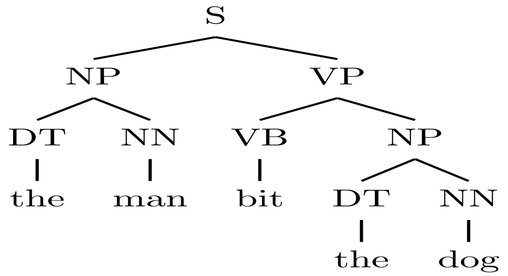


1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

Computational Linguistics => Computational Biology

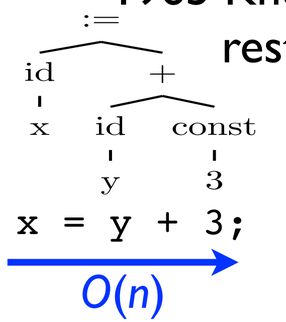
linguistics | **compiler theory** | **comp. linguistics** | **computational biology**

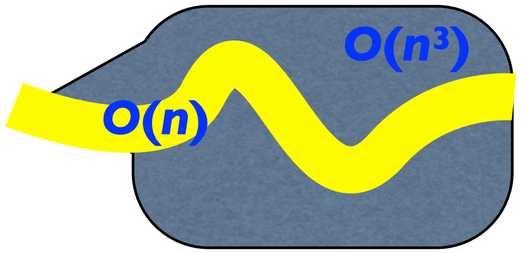
1955 Chomsky: context-free grammars (CFGs)



1958 Backus & Naur: CFGs for program, lang.

1965 Knuth: **LR** parsing for restricted CFGs: ***O(n)***

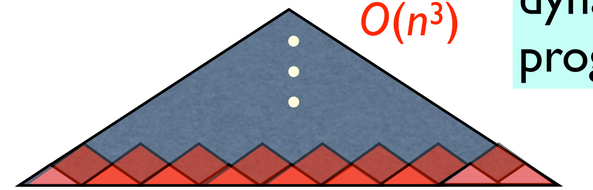




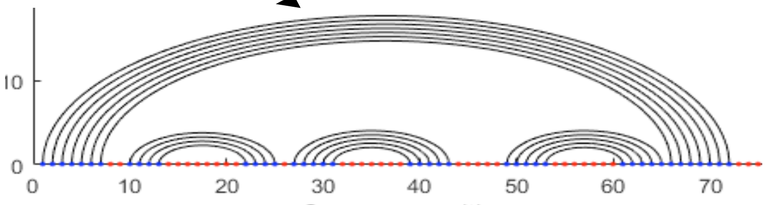
1964 **C**ocke \ bottom-up
1965 **K**asami - **CKY** ***O(n³)***
1967 **Y**ounger / for all CFGs

1986 Tomita: **G**eneralized **LR** for all CFGs: ***O(n³)***

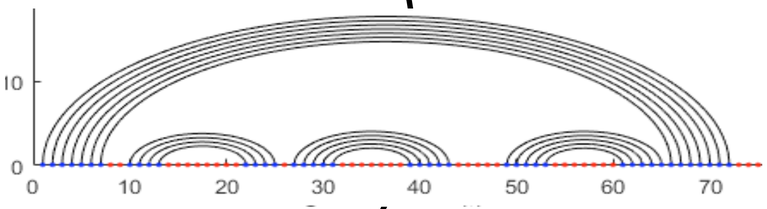
2010: Huang & Sagae: ***O(n)***
(approx.) DP for all CFGs



dynamic programming



1978: Nussinov ***O(n³)*** RNA folding
1981: Zuker & Siegler

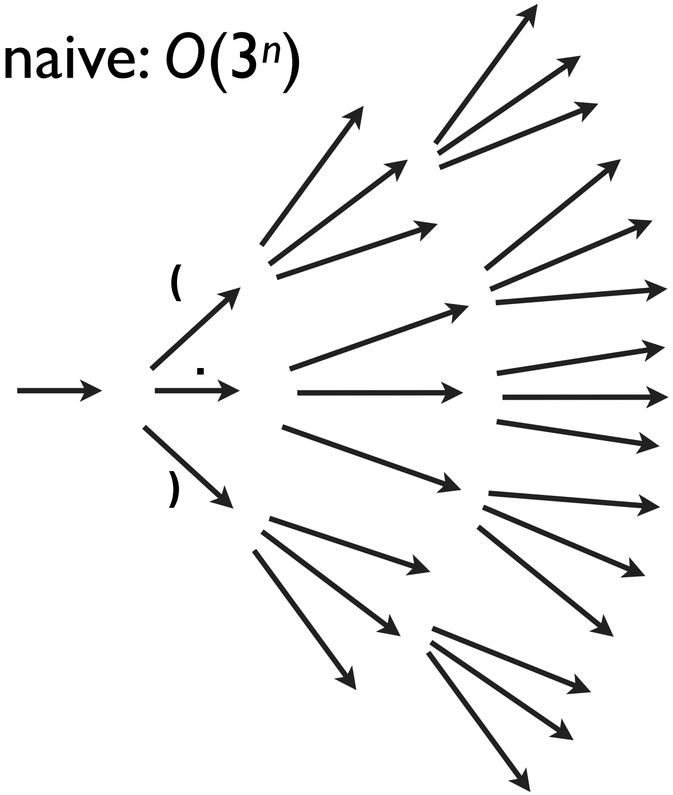


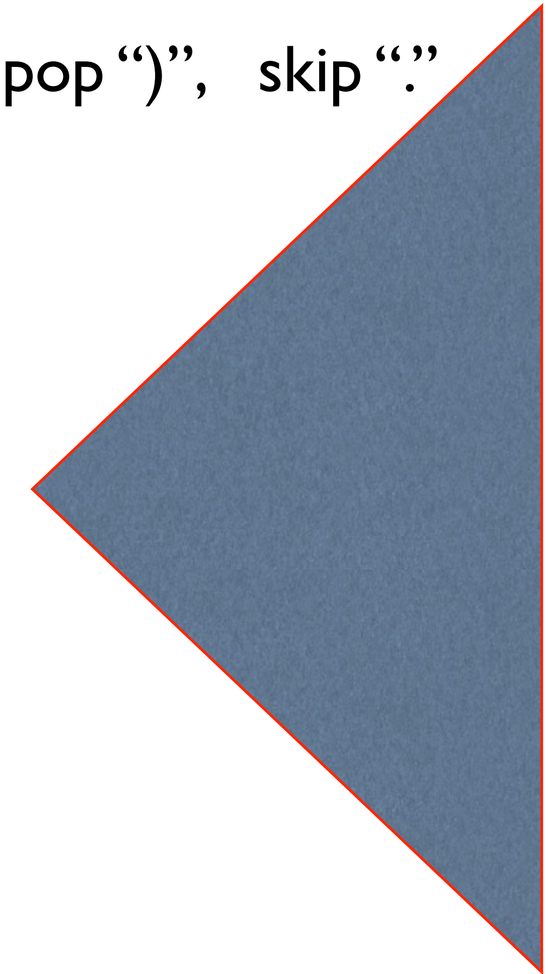
2019: LinearFold O(n)
(approx.) RNA folding

How to Fold RNAs in Linear-Time?

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

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

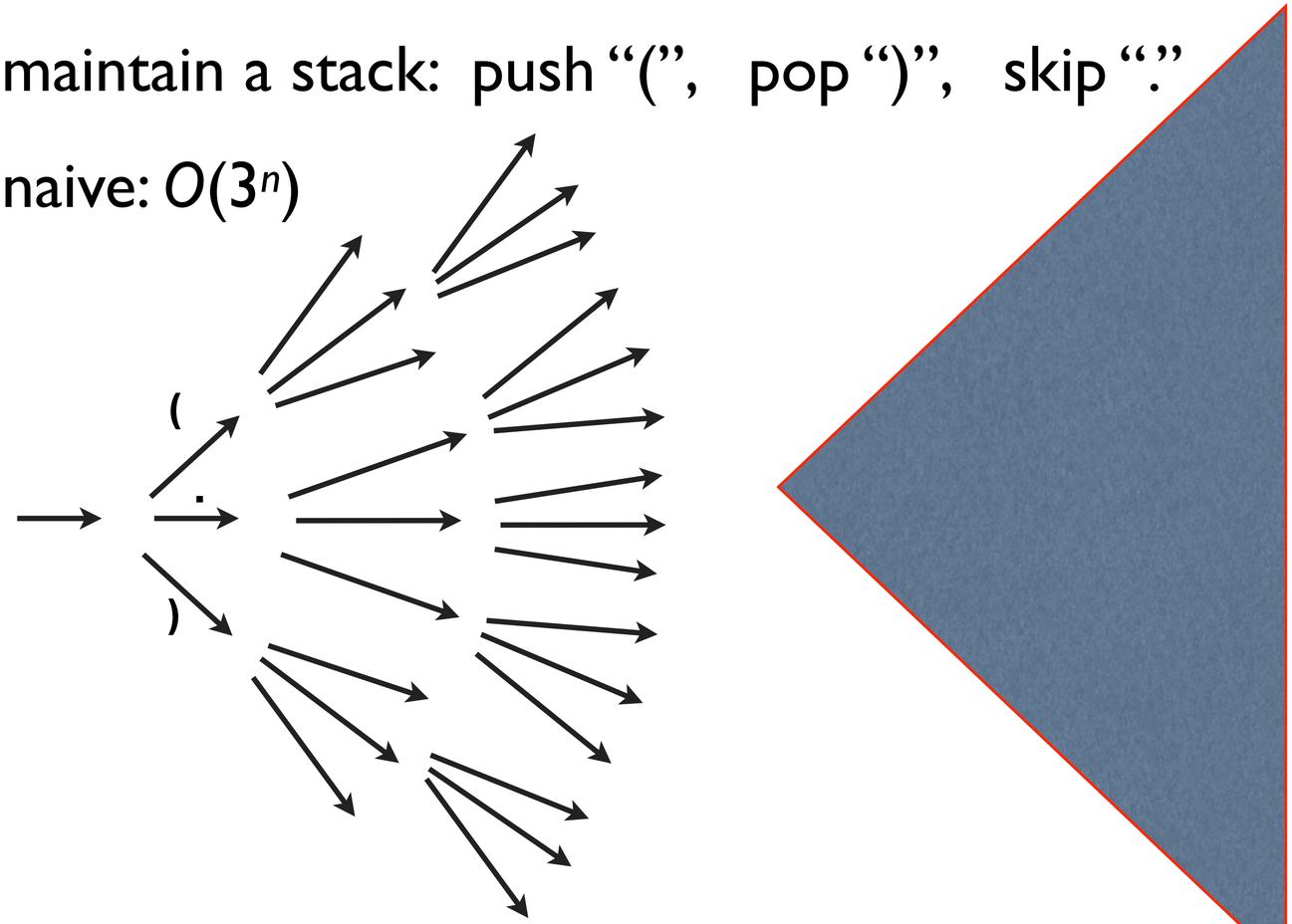




How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((( . . ((((. . . . . . . )))) . ((((. . . . . . )))) . . . . ((((. . . . . . )))))))))))) . . . .
```

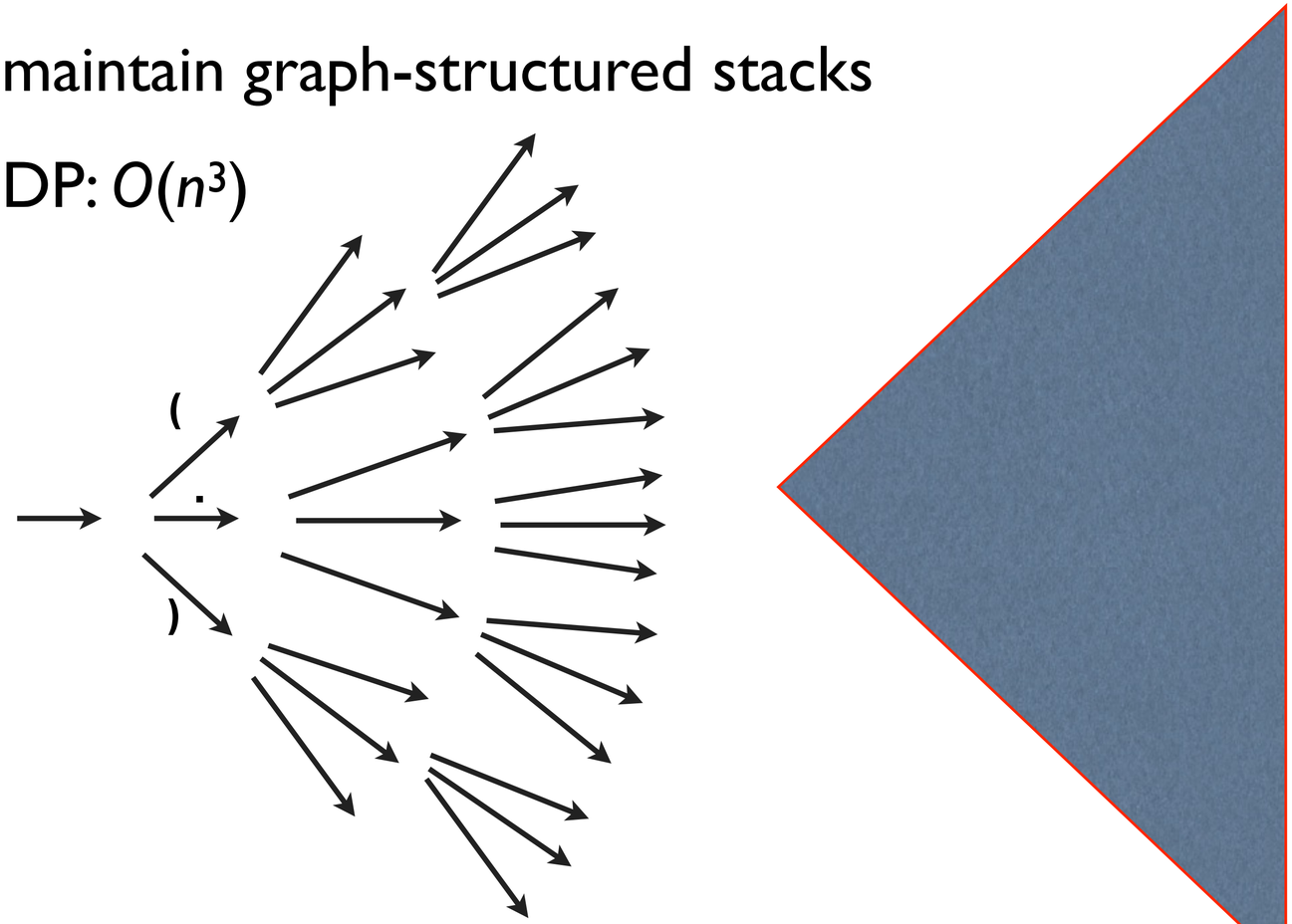
- idea 0: tag each nucleotide from left to right
  - maintain a stack: push "(", pop ")", skip "."
  - naive: $O(3^n)$



How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
(((((((..(((((.....)))))..(((((.....)))))).....(((((.....)))))))))))....
```

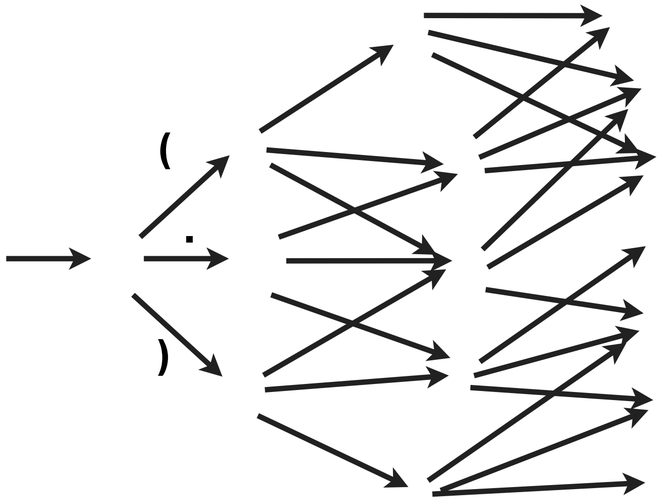
- idea 1: DP by packing "equivalent states"
  - maintain graph-structured stacks
  - DP: $O(n^3)$

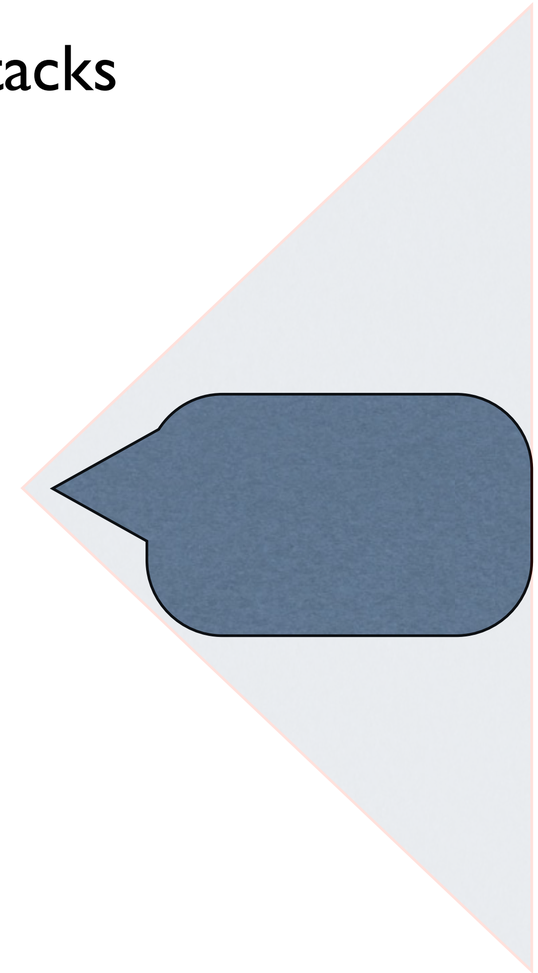


How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
(((((((..(((((.....)))))..(((((.....)))))).....(((((.....)))))))))))....
```

- idea 1: DP by packing "equivalent states"
  - maintain graph-structured stacks
  - DP: $O(n^3)$

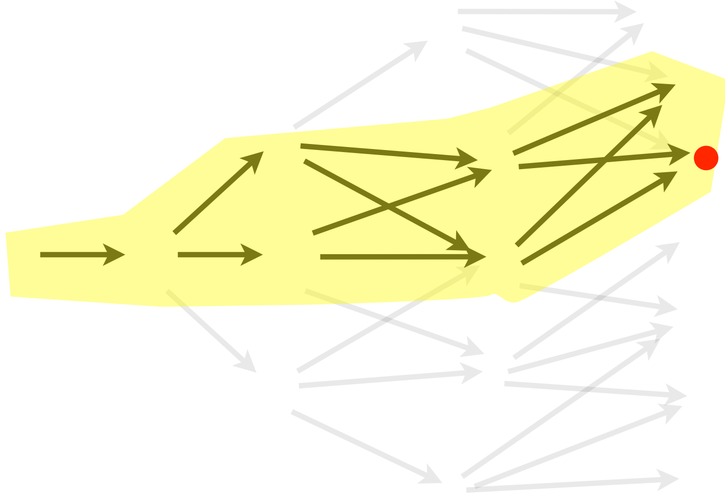


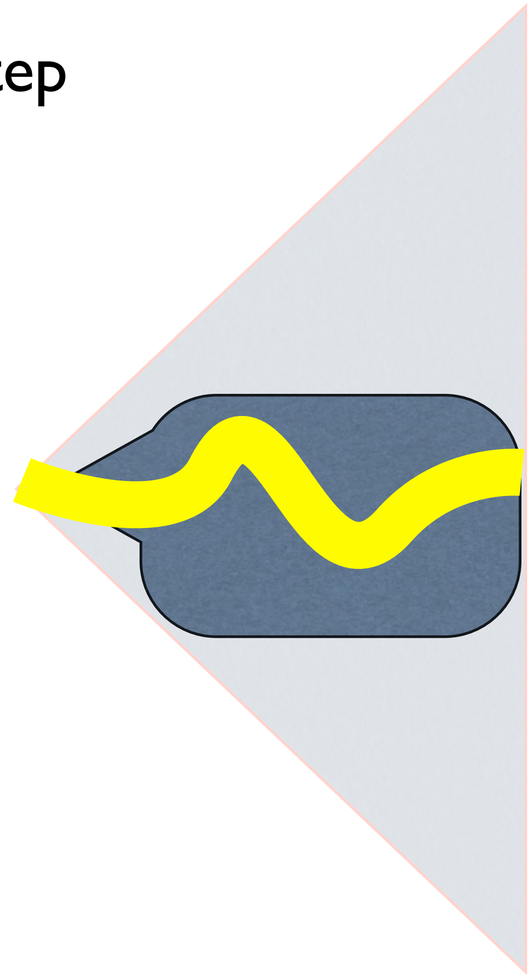


How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((((..((((.........)))).((((.........))))).....((((.........))))))))))....
```

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

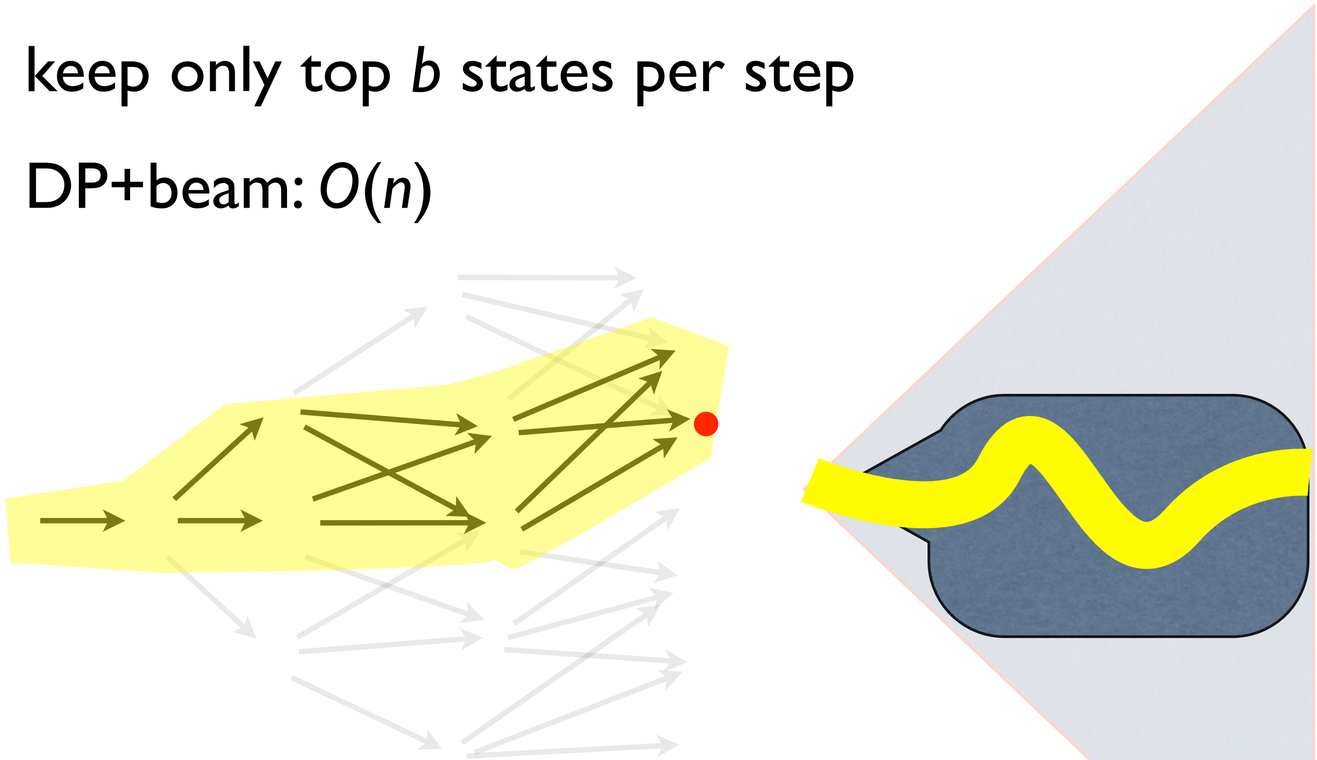




How to Fold RNAs in Linear-Time?

```
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA
((((((( . . ((((. . . . . . . )))) . ((((. . . . . . )))) . . . . ((((. . . . . . ))))))))))))) . . . .
```

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



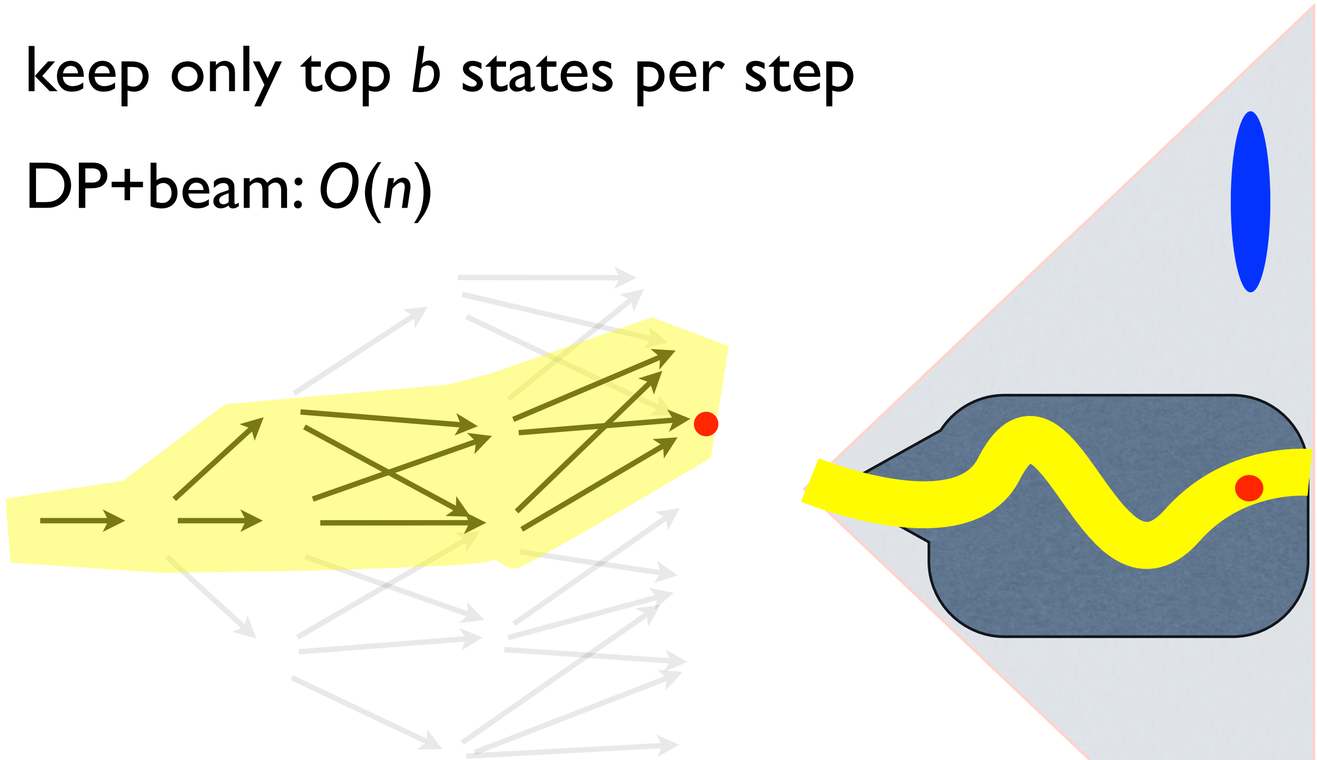
each DP state corresponds to exponentially many non-DP states

How to Fold RNAs in Linear-Time?

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

(((((((. . ((((. . . . .)))) . ((((. . . . .)))) . . . . ((((. . . . .)))))))))))) . . . .

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



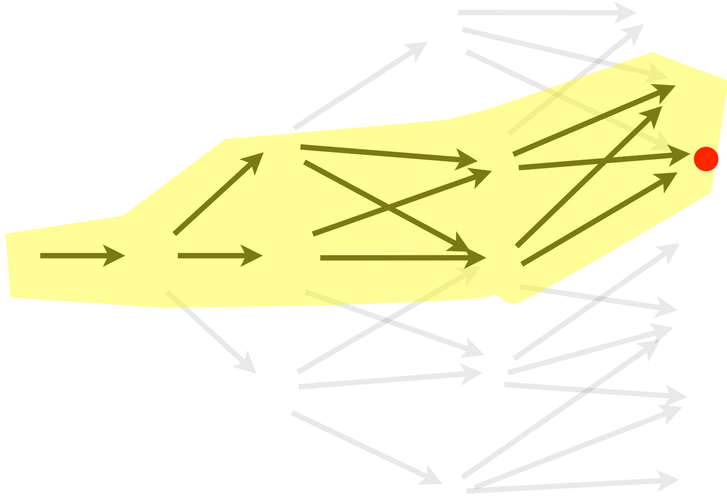
each DP state corresponds to exponentially many non-DP states

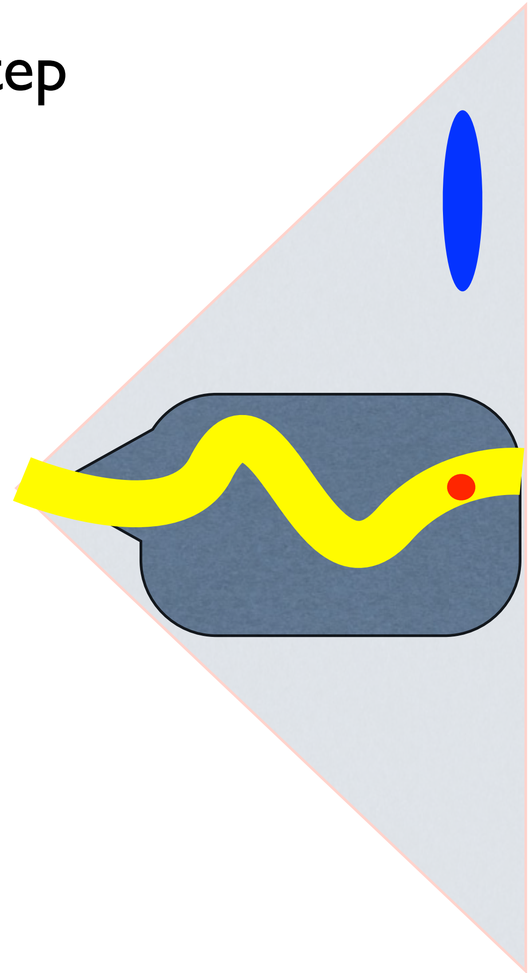
How to Fold RNAs in Linear-Time?

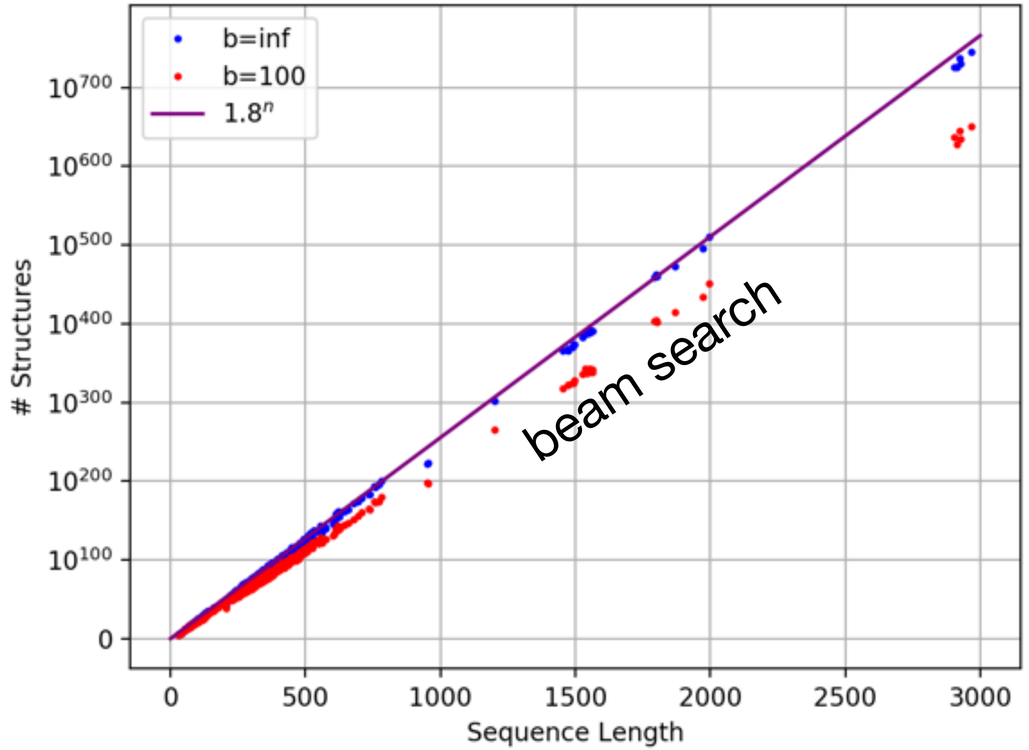
GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

(((((((. . ((((. . . . .)))) . ((((. . . . .)))) . . . . ((((. . . . .)))))))))))) . . .

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



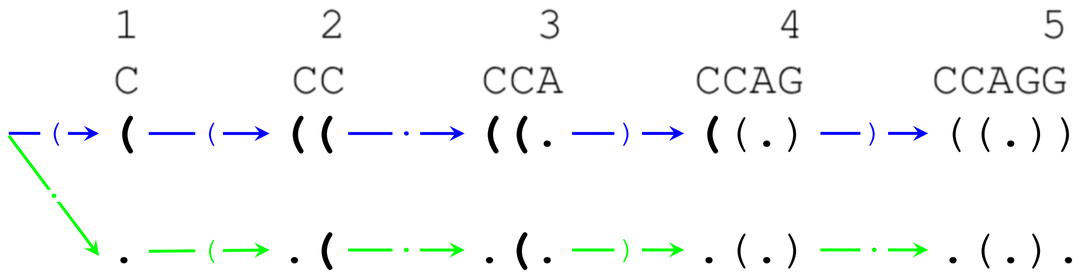


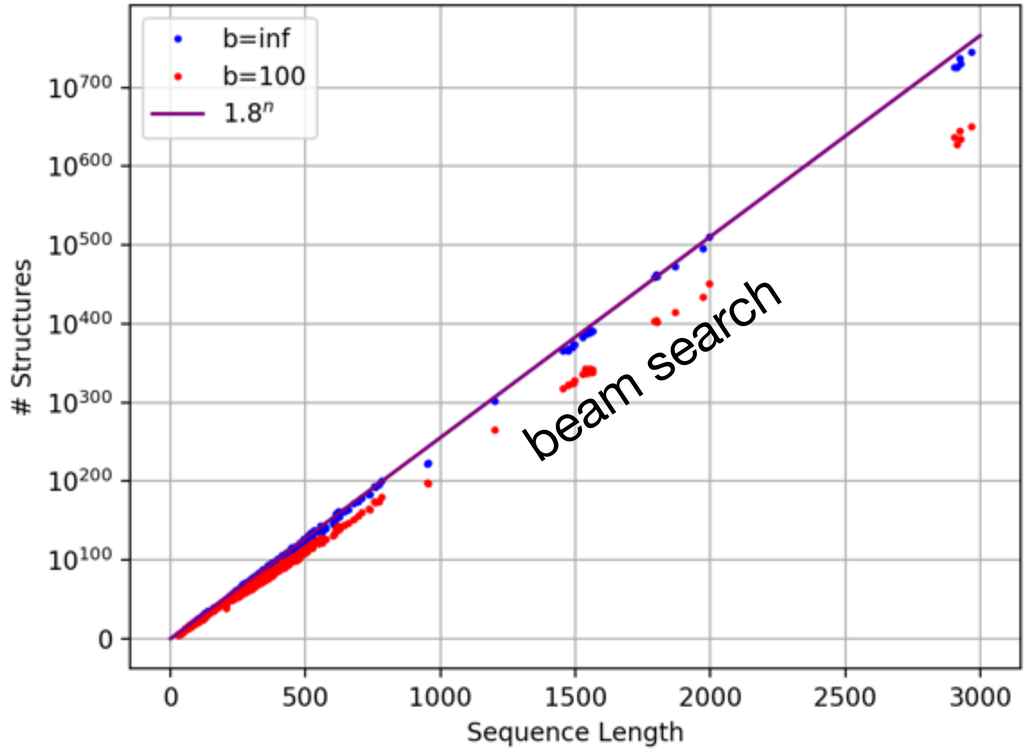


each DP state corresponds to exponentially many non-DP states

Details: Packing "Temporarily Equivalent" States

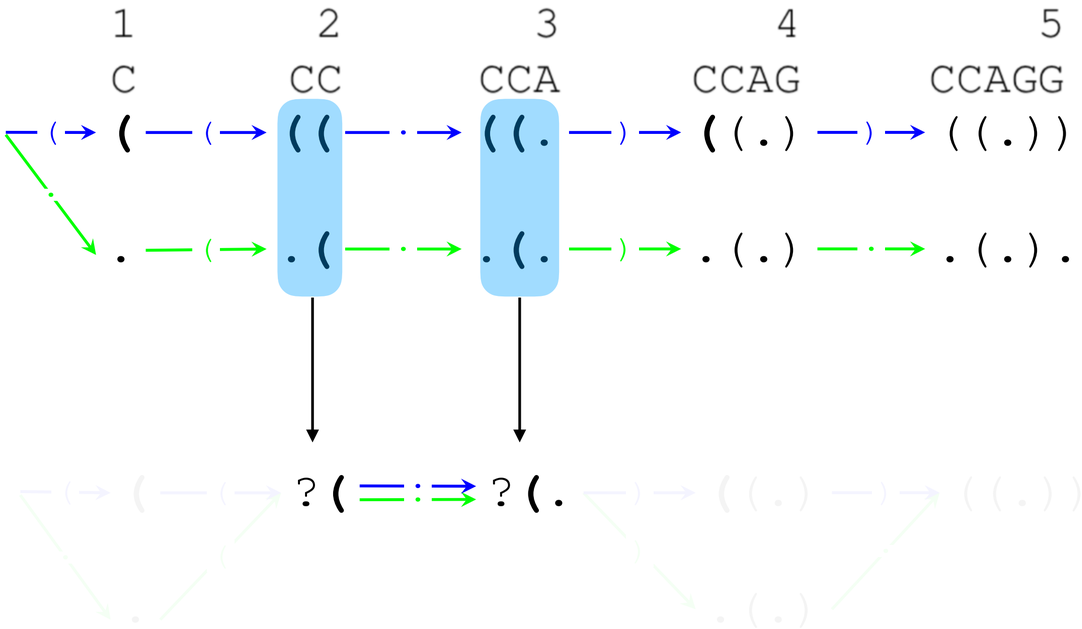
- two states are "temporarily equivalent" if they share the same stack top index
  - because they are looking for the same nucleotide(s) to match this stack-top nucleotide
  - we pack them as a single state until a matching is found, when they unpack
  - this is how we can explore exponentially many structures in linear time

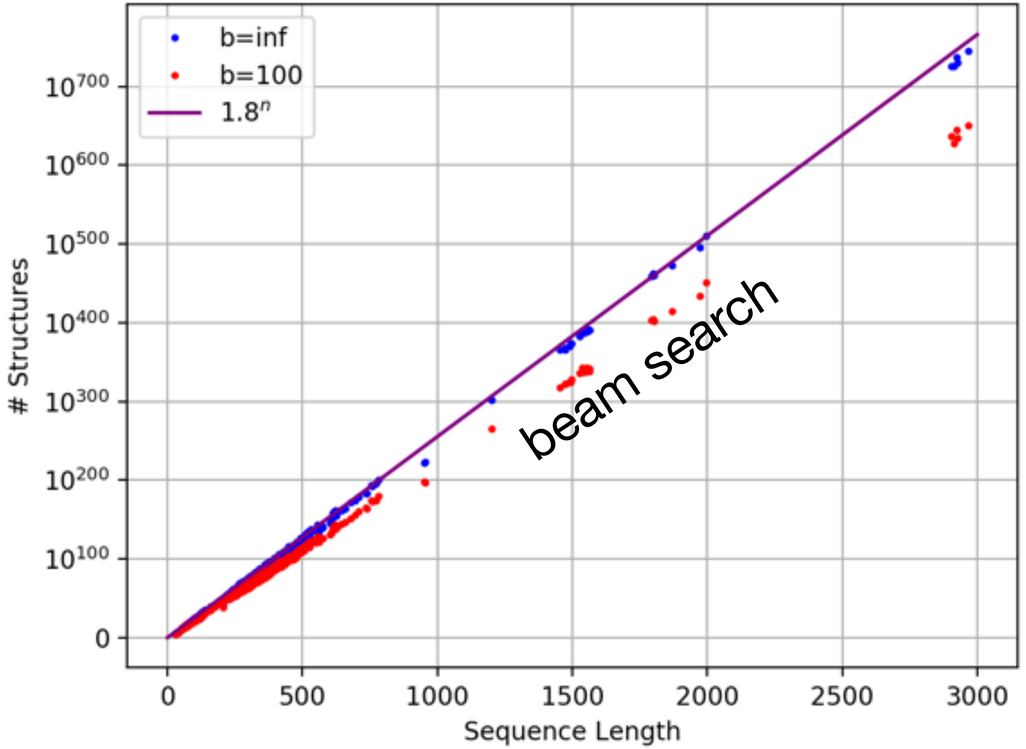




Details: Packing "Temporarily Equivalent" States

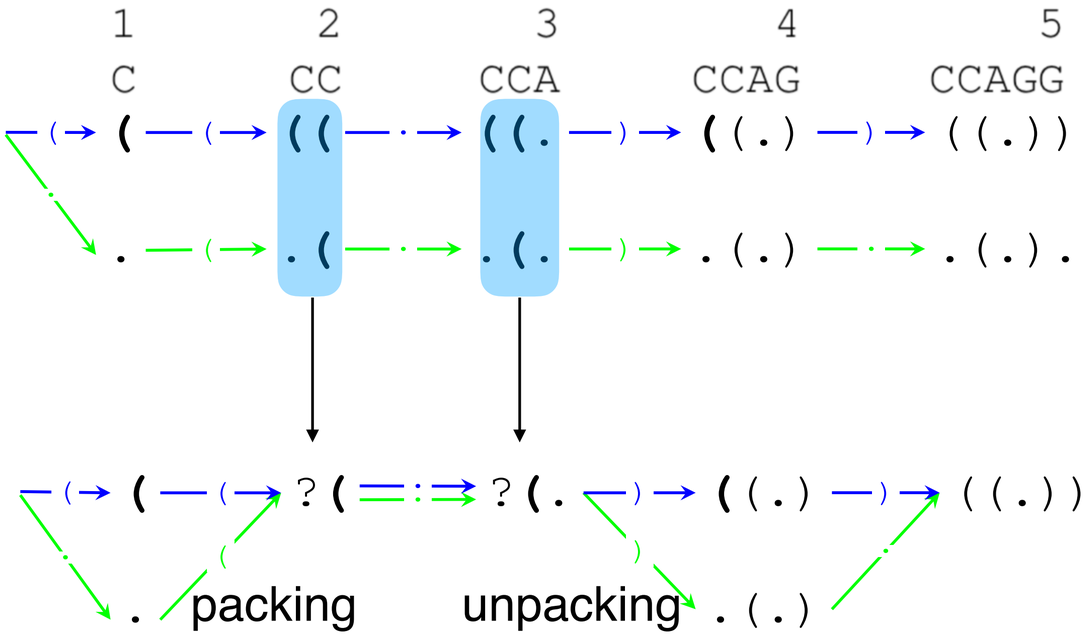
- two states are "temporarily equivalent" if they share the same stack top index
  - because they are looking for the same nucleotide(s) to match this stack-top nucleotide
  - we pack them as a single state until a matching is found, when they unpack
  - this is how we can explore exponentially many structures in linear time

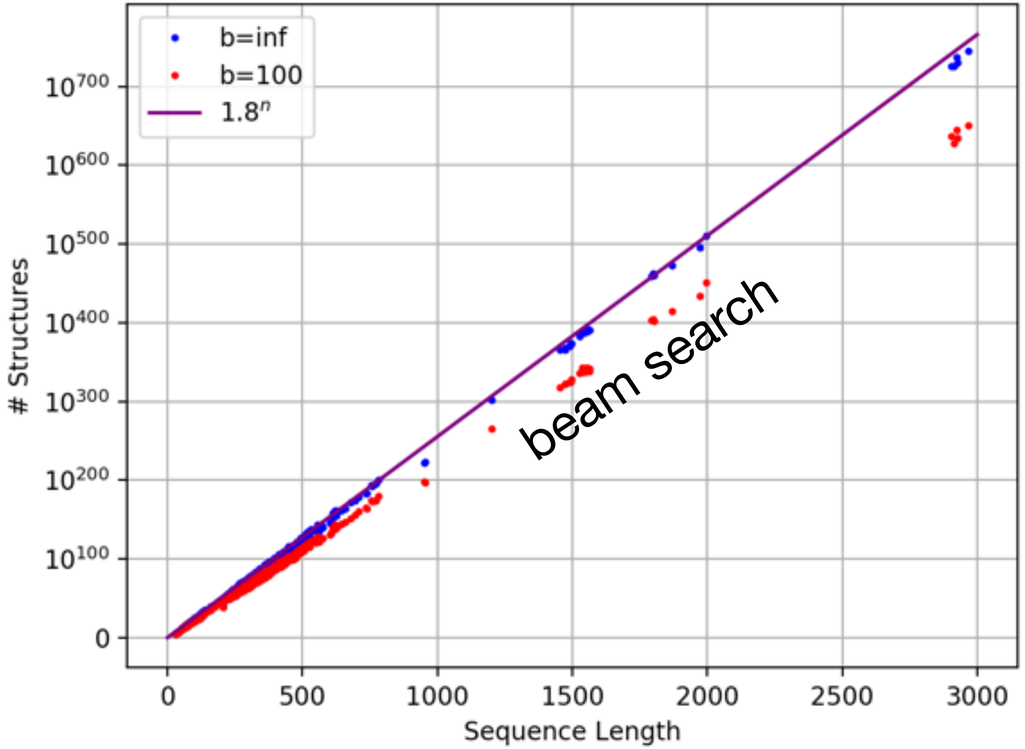




Details: Packing "Temporarily Equivalent" States

- two states are "temporarily equivalent" if they share the same stack top index
  - because they are looking for the same nucleotide(s) to match this stack-top nucleotide
  - we pack them as a single state until a matching is found, when they unpack
  - this is how we can explore exponentially many structures in linear time

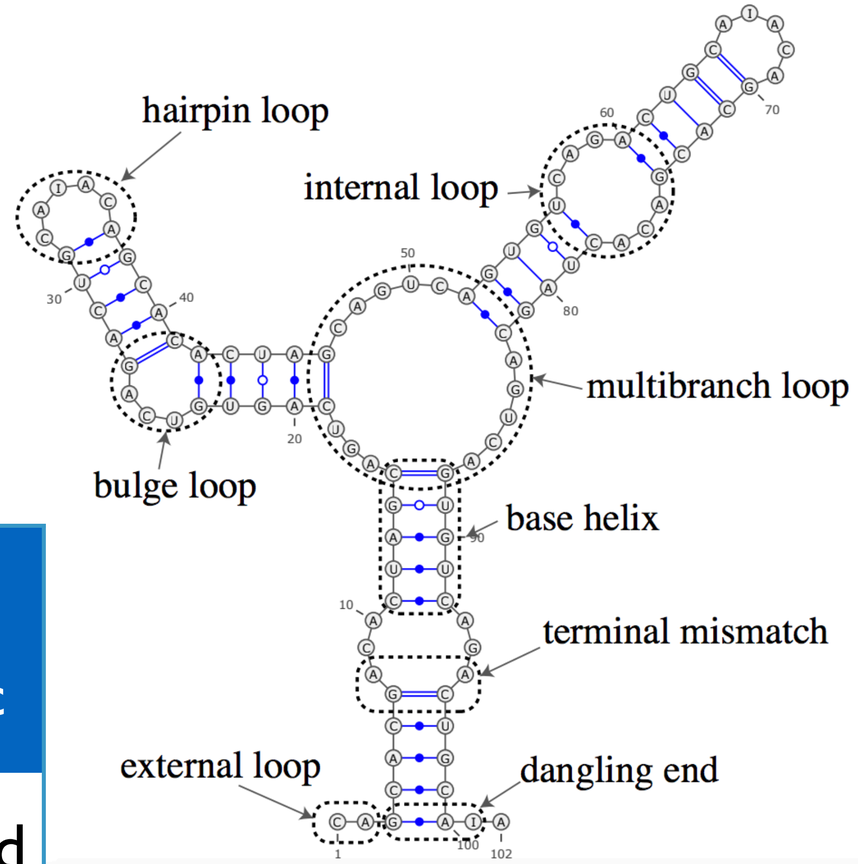




Results

LinearFold with SOTA Prediction Models

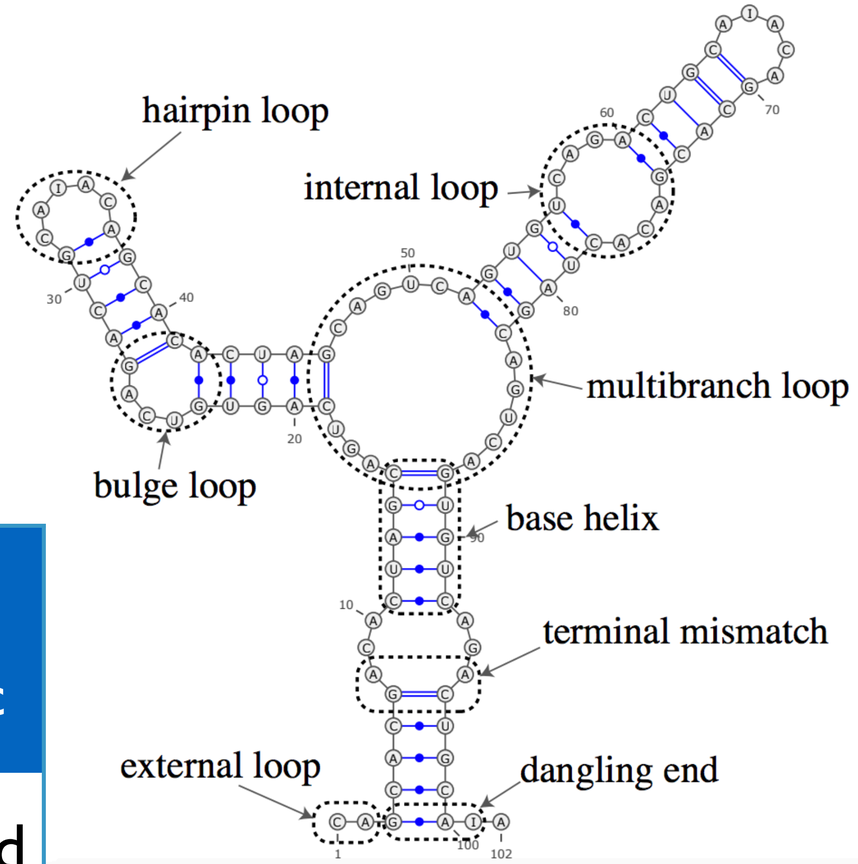
- models from two widely-used folding engines
  - CONTRAfold MFE (machine-learned)
  - Vienna RNAfold (thermodynamic)
- we linearized both systems from $O(n^3)$ to $O(n)$



	efficiency		*systems*	
	time	space	machine-learned	thermo-dynamic
baselines	$O(n^3)$	$O(n^2)$	**C**ONTRAfold	**V**ienna RNAfold
our work	$O(n)$	$O(n)$	LinearFold-**C**	LinearFold-**V**

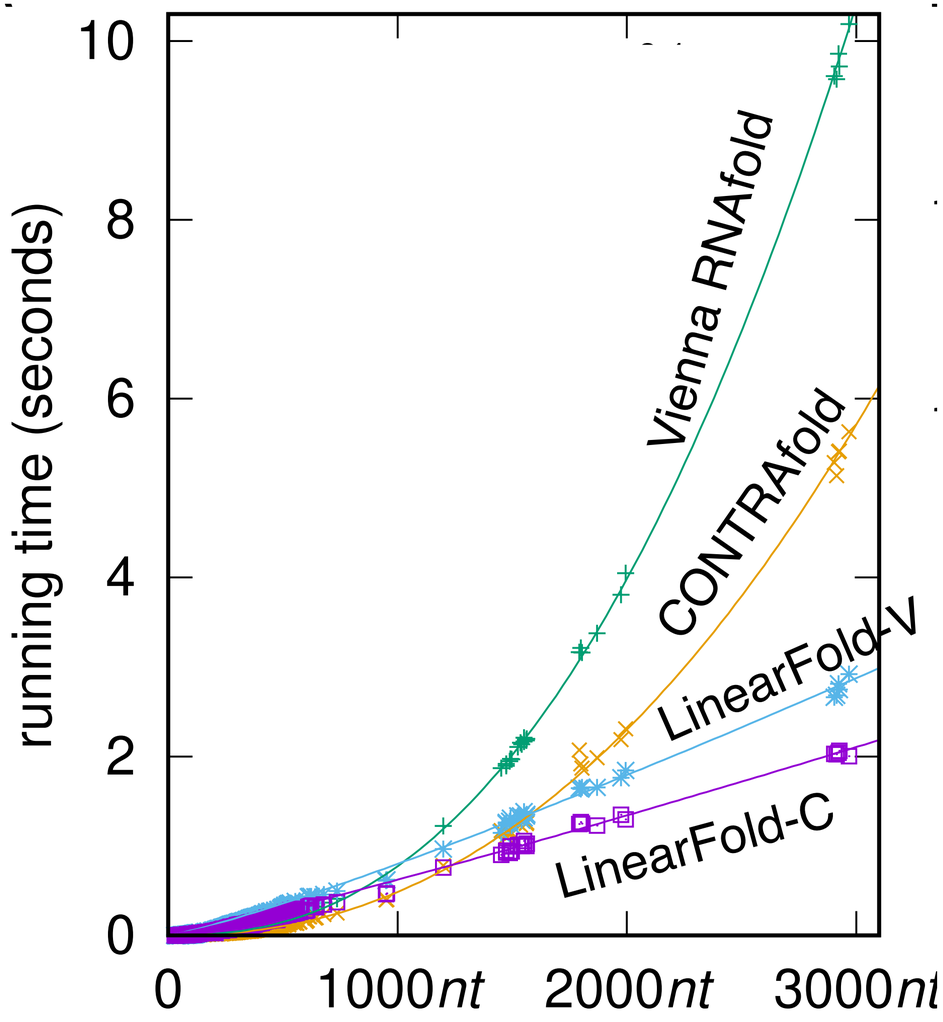
LinearFold with SOTA Prediction Models

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

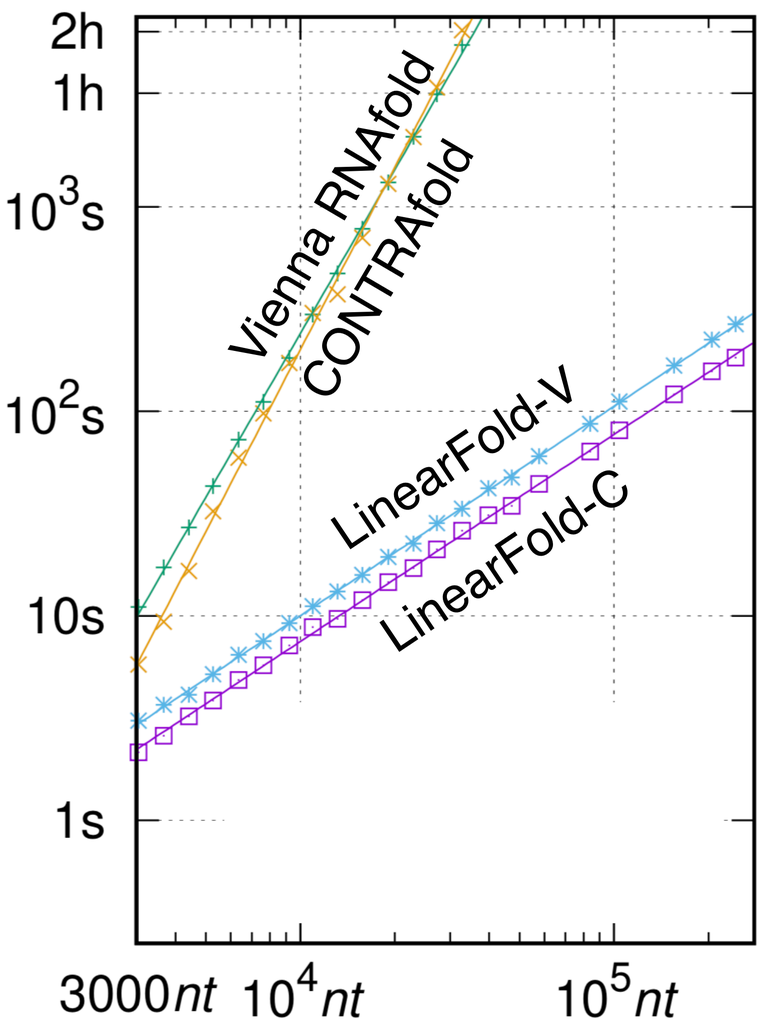


	efficiency		*systems*	
	time	space	machine-learned	thermo-dynamic
baselines	$O(n^3)$	$O(n^2)$	**C**ONTRAfold	**V**ienna RNAfold
our work	$O(n)$	$O(n)$	LinearFold-**C**	LinearFold-**V**

Efficiency & Scalability: *O*(*n*) time, *O*(*n*) memory



Archive II data set
(~3,000 seqs, max len: ~3,000 *nt*)



RNAcentral data set
(sampled, max len: ~250,000 *nt*)

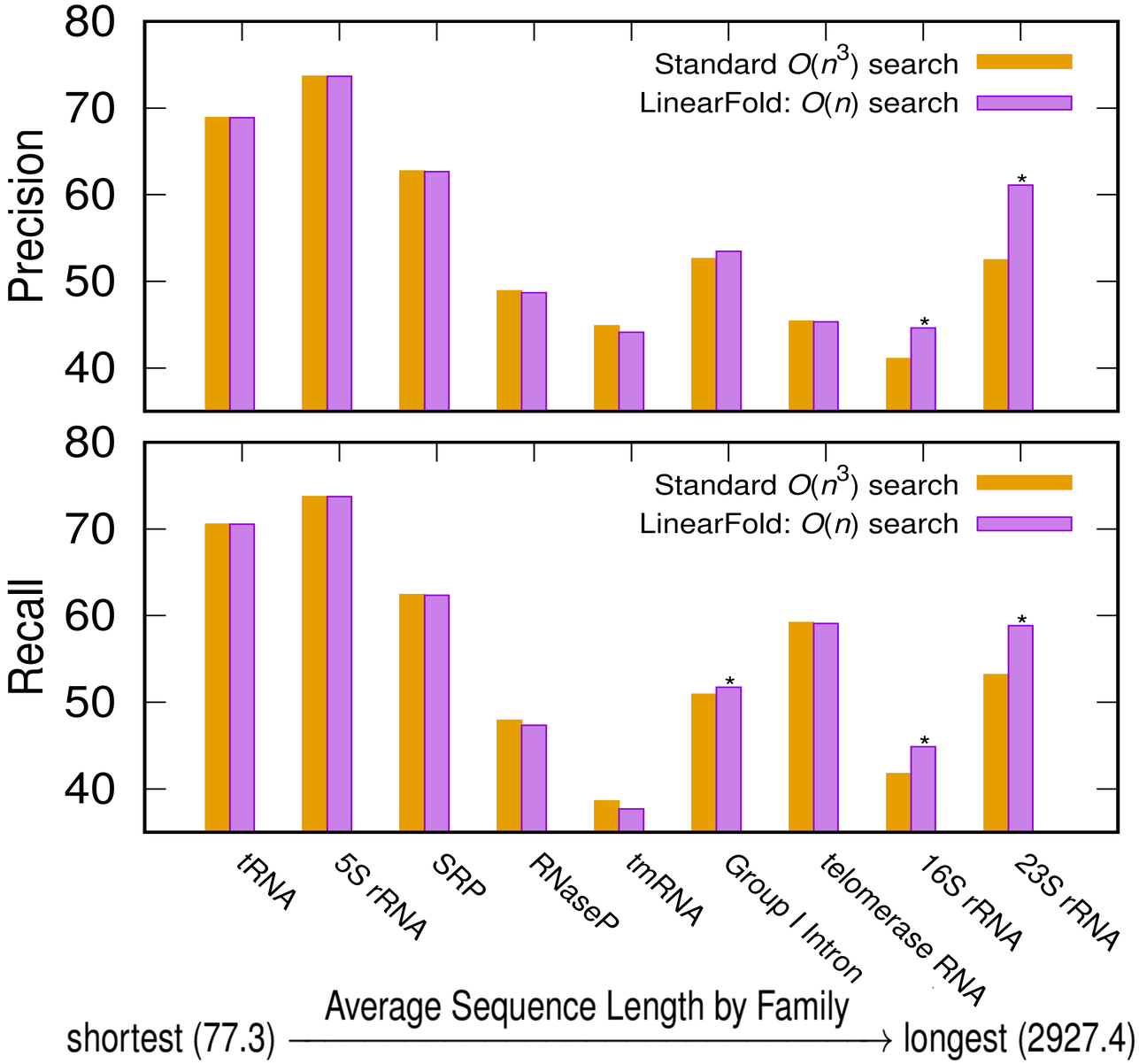
10,000*nt* (~HIV)
4min → 7s

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

Accuracy

- Tested on Archive II dataset (on a family-by-family basis)
  - significantly better on 3 long families
  - biggest boost on the longest families: 16S/23S rRNAs
- LinearFold-V vs. Vienna is similar

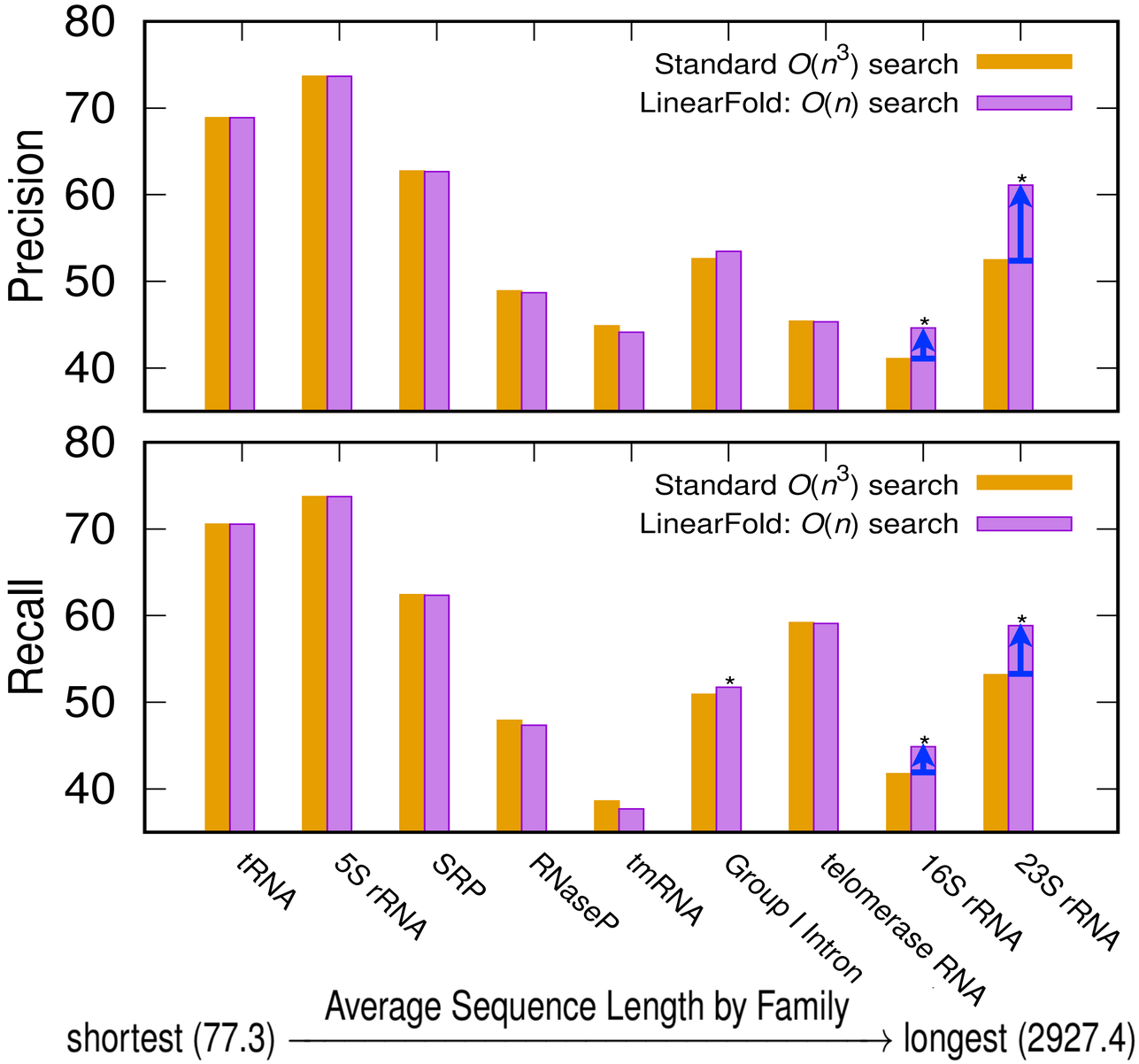
Overall	(precision) PPV		(recall) Sensitivity	
CONTRAfold MFE	54.51		55.36	
LinearFold-C	55.84	(+1.3)	56.24	(+0.9)
Vienna RNAfold	50.22		58.74	
LinearFold-V	50.51	(+0.3)	58.97	(+0.2)



Accuracy

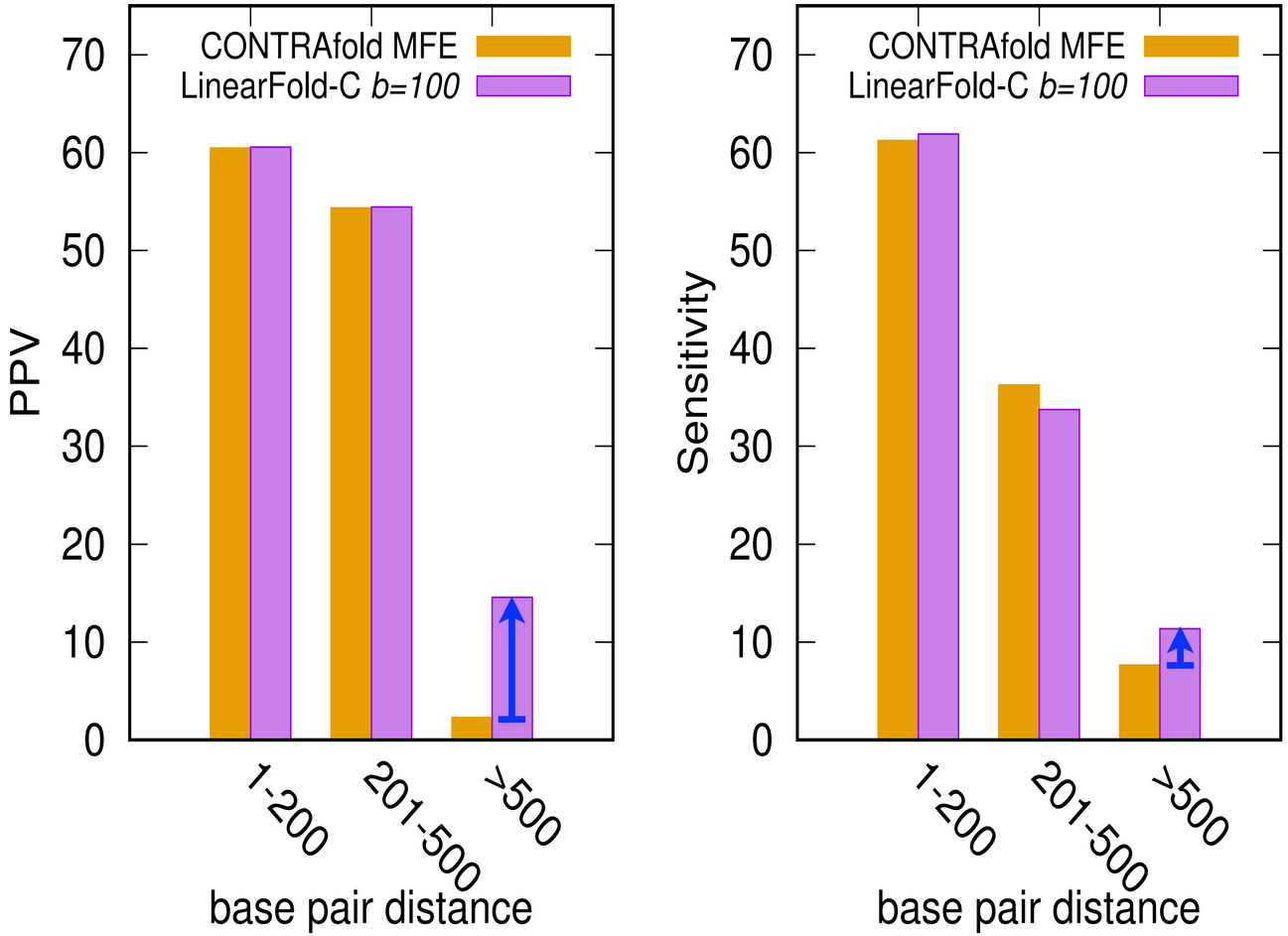
- Tested on Archive II dataset (on a family-by-family basis)
  - significantly better on 3 long families
  - biggest boost on the longest families: 16S/23S rRNAs
- LinearFold-V vs. Vienna is similar

	(precision)		(recall)	
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)



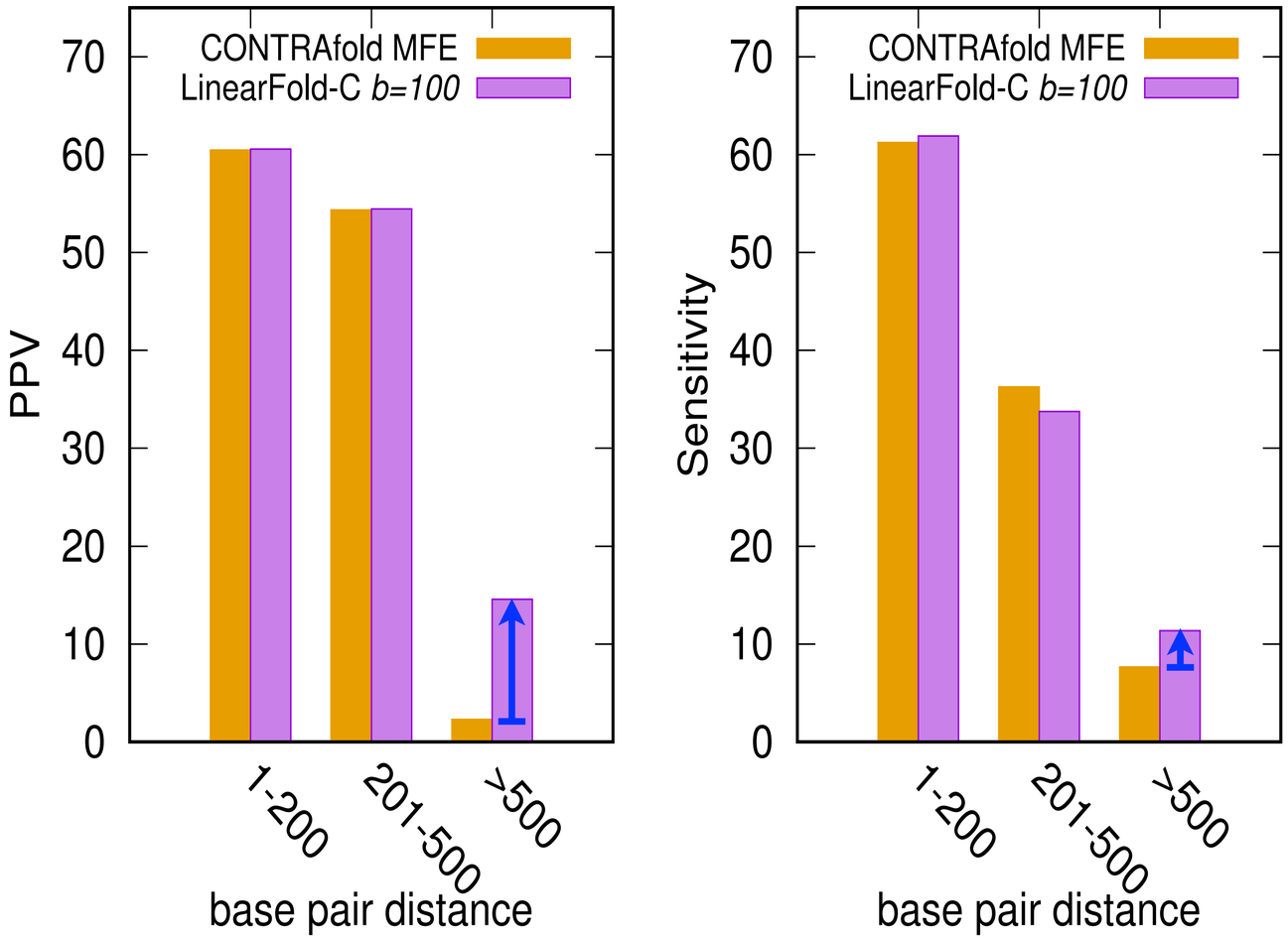
Improvements on Long-Range Base Pairs

- long-distance pairs are well-known to be hard to predict
  - LinearFold outputs less long-range base pairs, but more correct ones

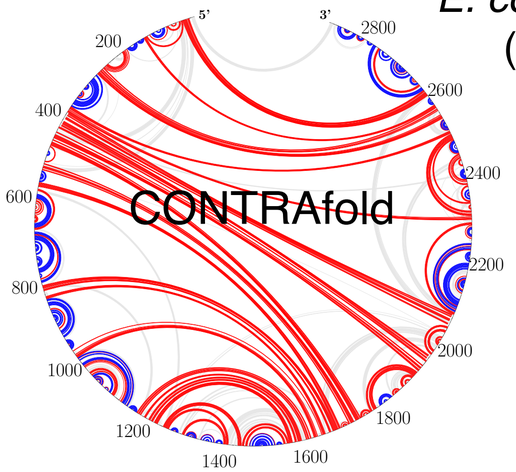


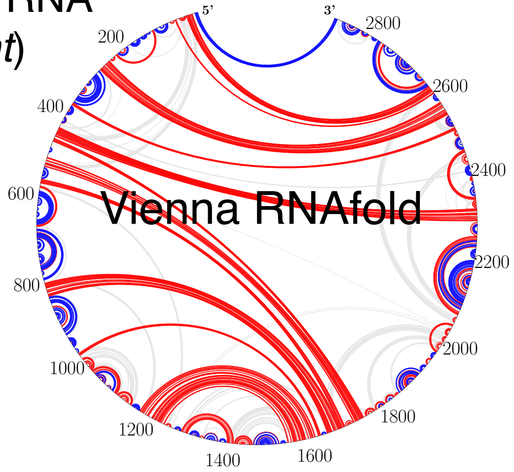
Improvements on Long-Range Base Pairs

- long-distance pairs are well-known to be hard to predict
  - LinearFold outputs less long-range base pairs, but more correct ones



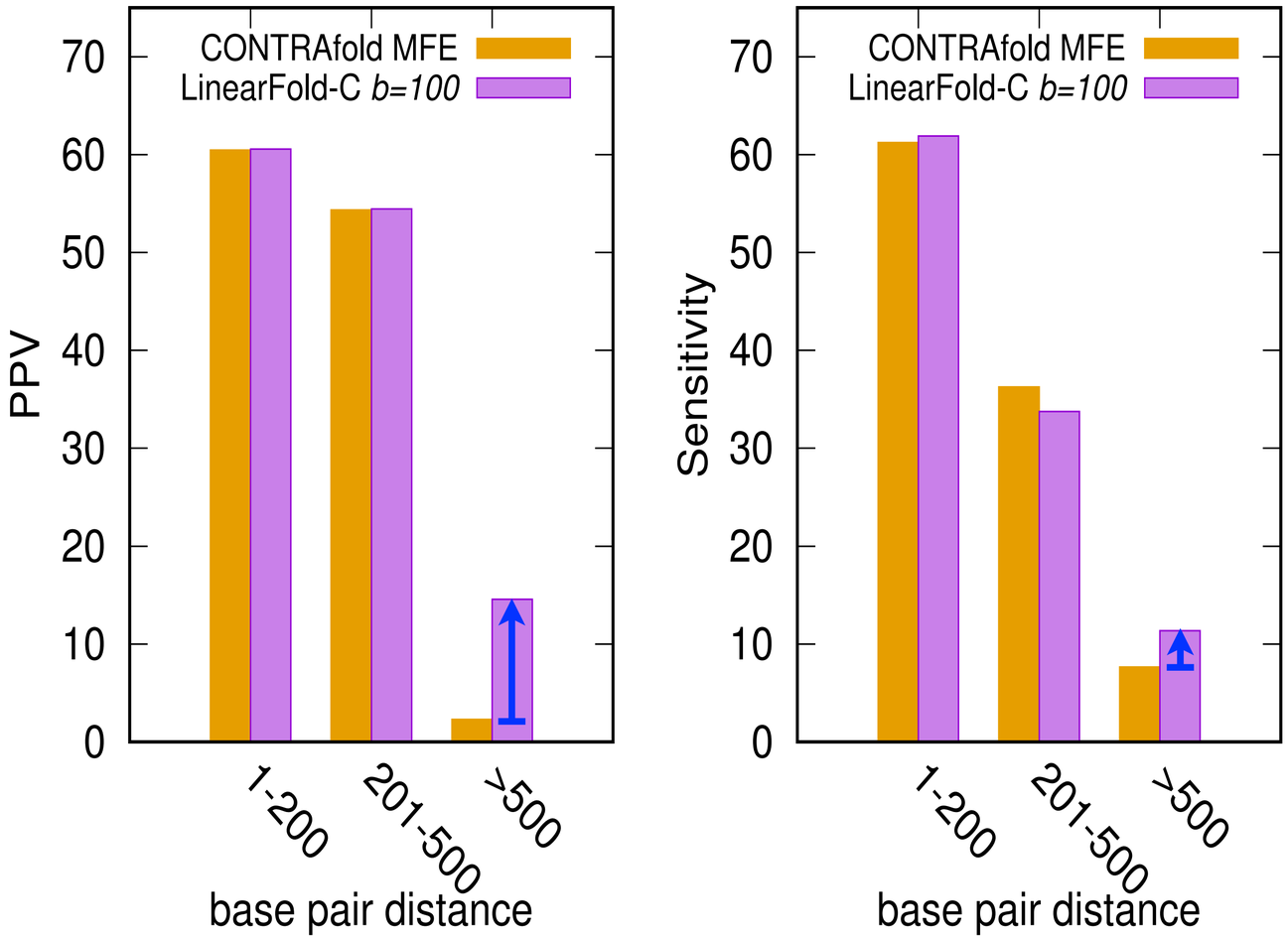
E. coli 23S rRNA
(2,904 *nt*)



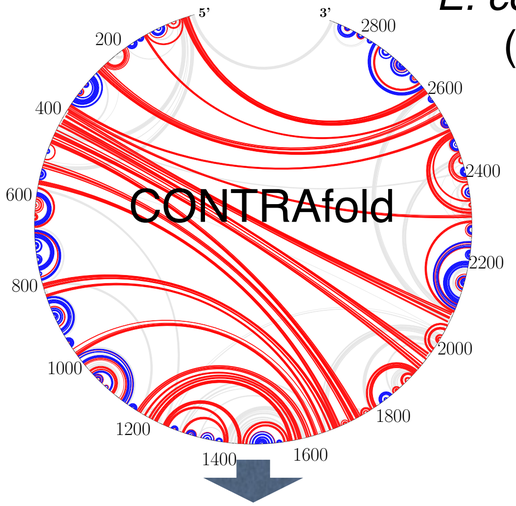


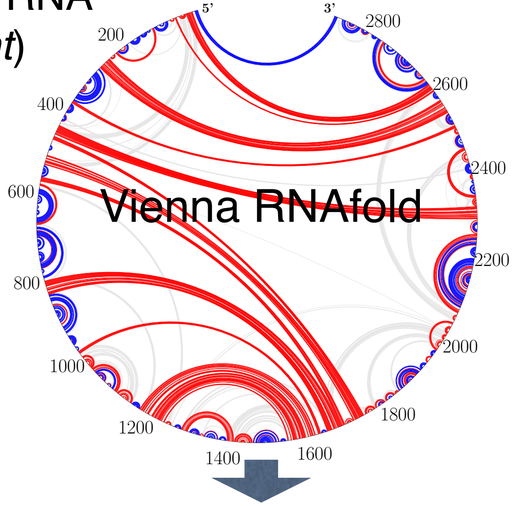
Improvements on Long-Range Base Pairs

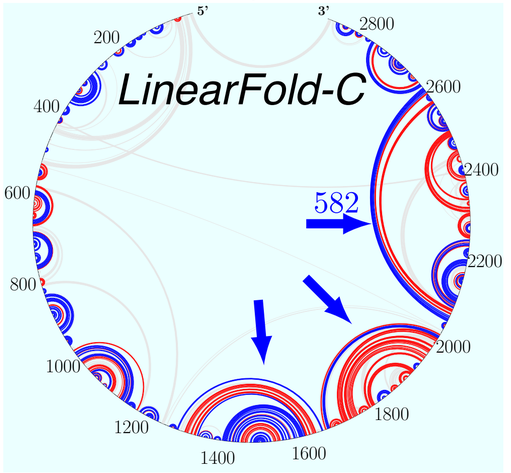
- long-distance pairs are well-known to be hard to predict
  - LinearFold outputs less long-range base pairs, but more correct ones

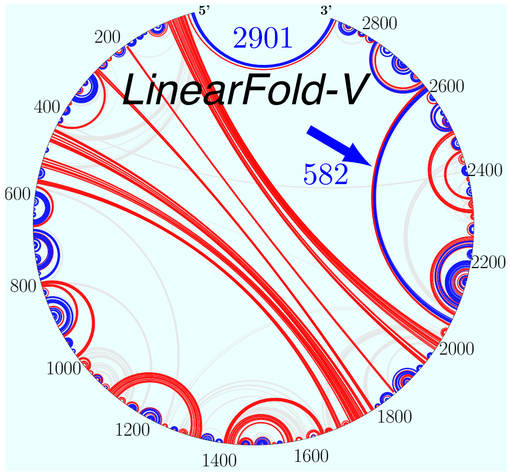


E. coli 23S rRNA
(2,904 *nt*)



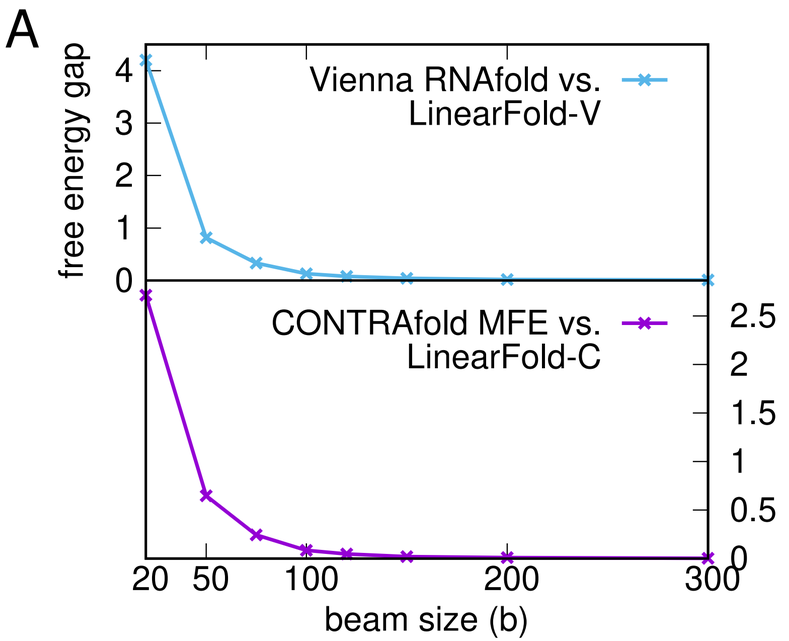


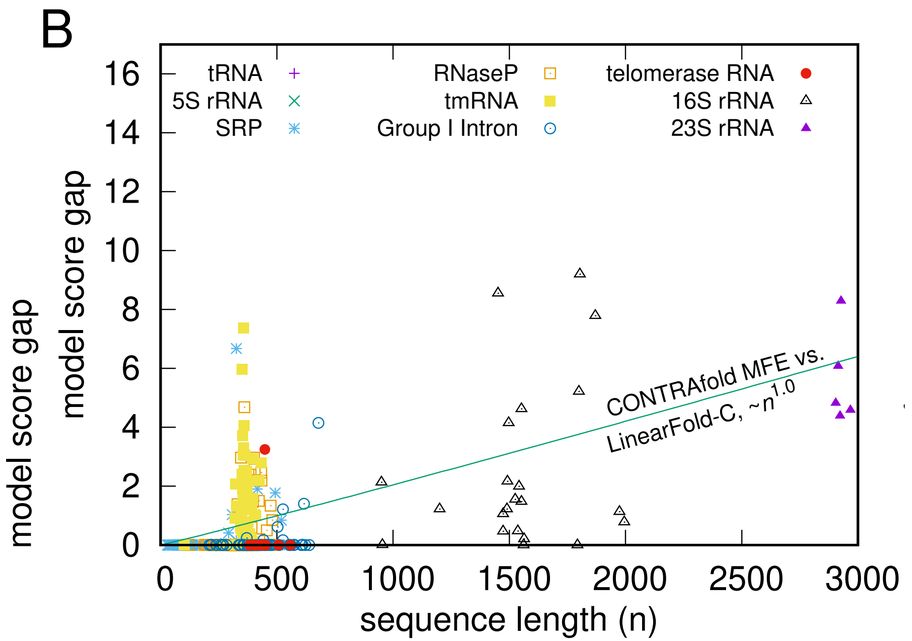


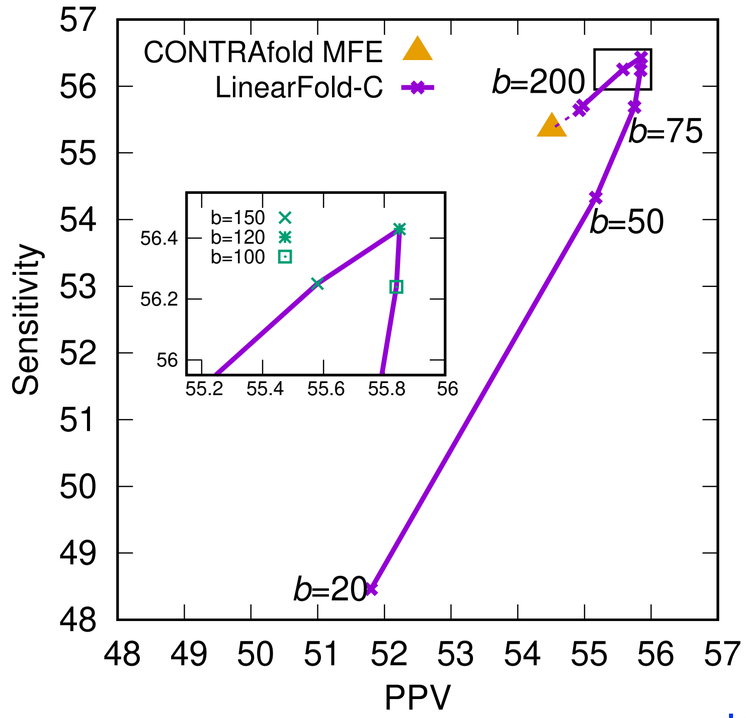


Search (Approximation) Quality

- search error = energy (or model score) gap between exact search and our search
  - search error quickly shrinks to 0 as beam size increases
  - search error grows linearly with sequence length (constant search error per nucleotide)
  - both PPV & Sensitivity peaks around beam=120, but we choose beam=100 for simplicity

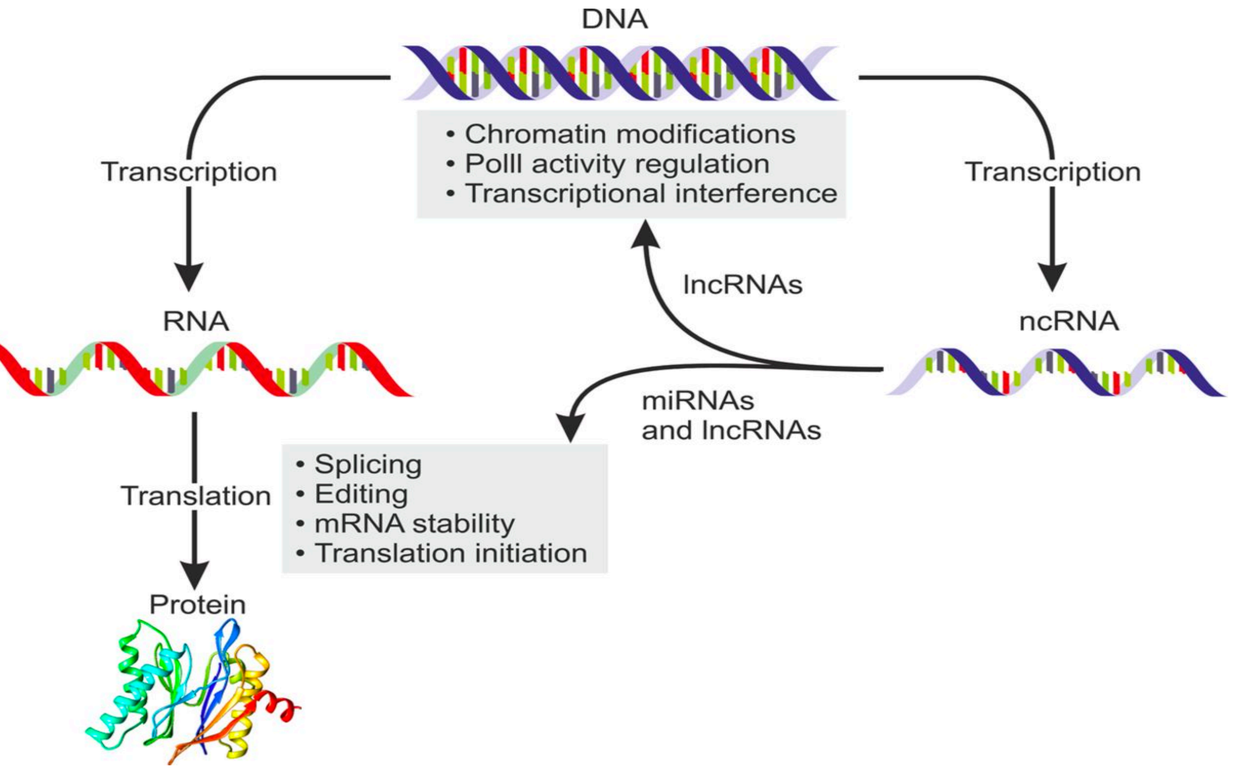


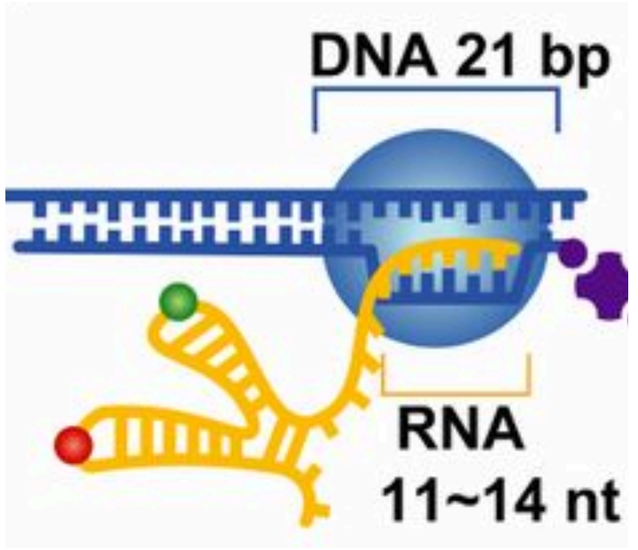


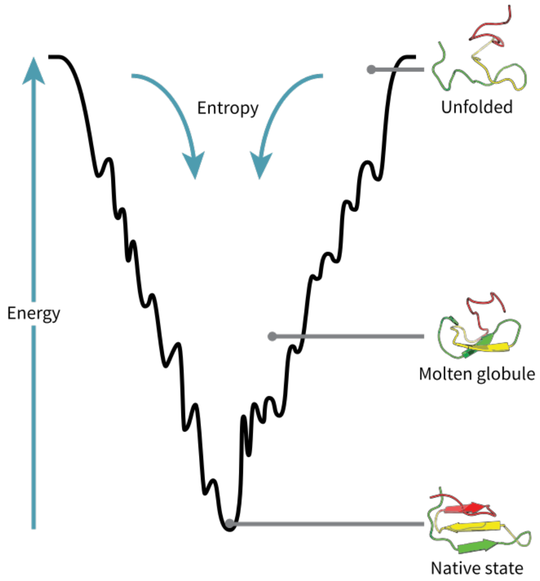


Connections to Cotranscriptional Folding

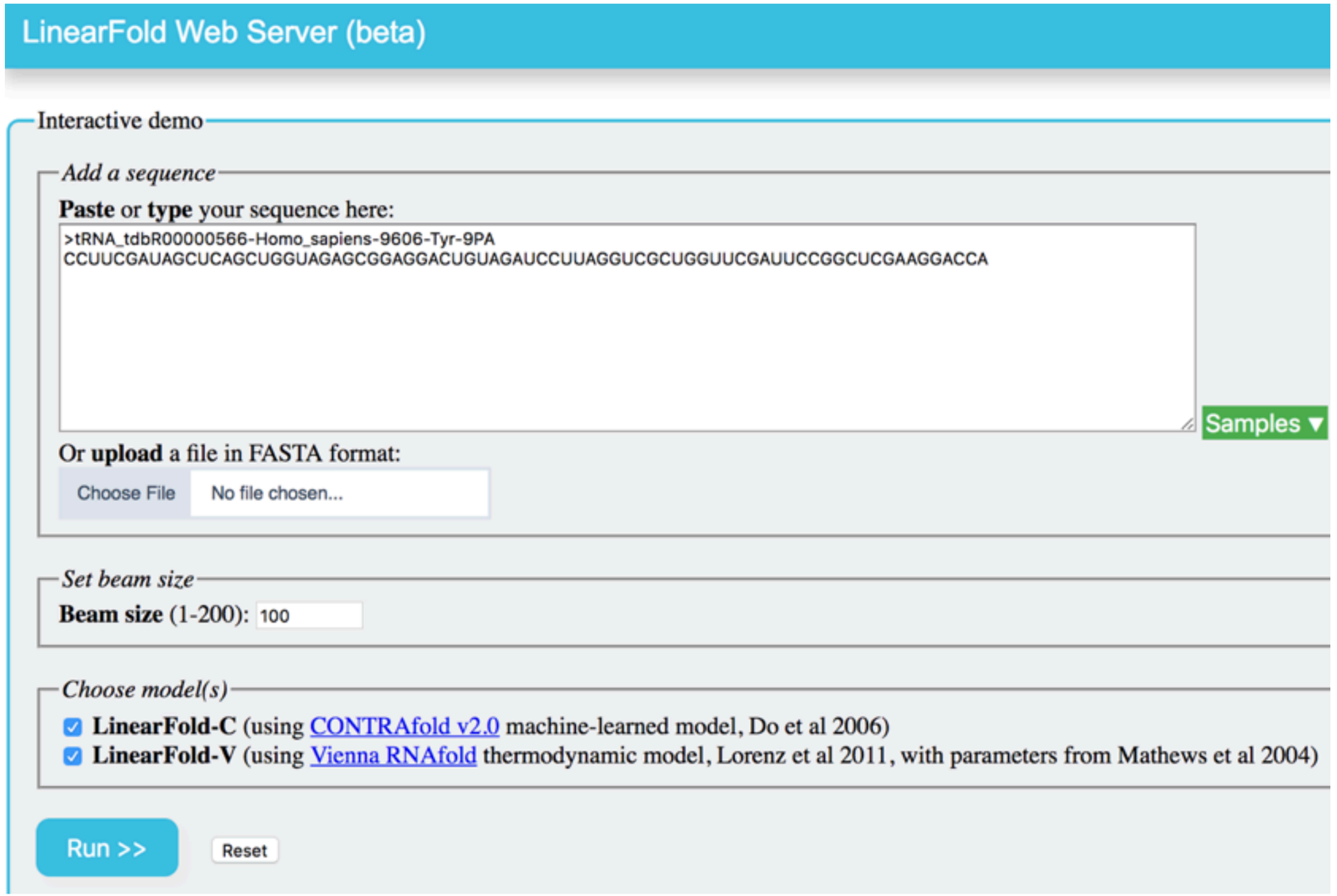
- RNAs & proteins fold while being assembled
- RNA & protein sequences evolve to be incrementally foldable
- these might explain why LinearFold performs better than exact search







LinearFold Server & GitHub

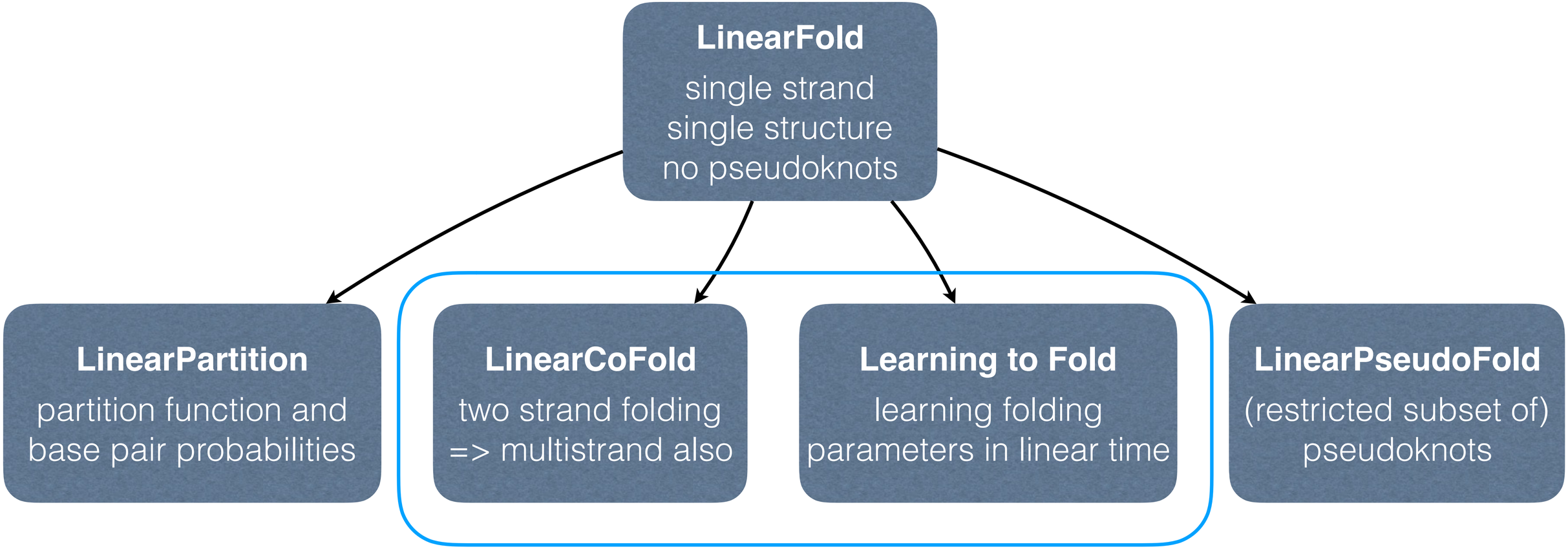


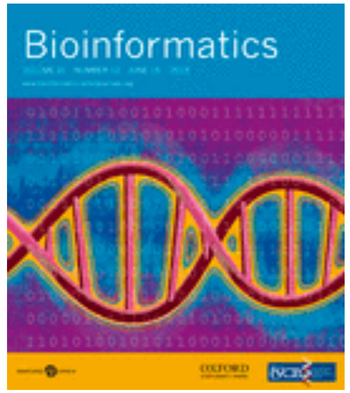
- fastest RNA folding server (by a very large margin)
  - thanks to O(n) time
- longest sequence limit
  - 100,000 nt (10x Vienna)
  - thanks to O(n) space
- source code on GitHub
  - unified implementation of LinearFold-C & LinearFold-V

http://linearfold.org

https://github.com/LinearFold/LinearFold

Extensions of LinearFold



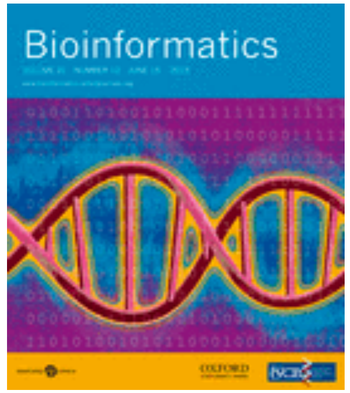


first linear-time (approximate) RNA folding algorithm

better in accuracy (esp. long seqs and long-range pairs)

http://linearfold.org

Thank you very much !



first linear-time (approximate) RNA folding algorithm

better in accuracy (esp. long seqs and long-range pairs)

http://linearfold.org

Oregon State University

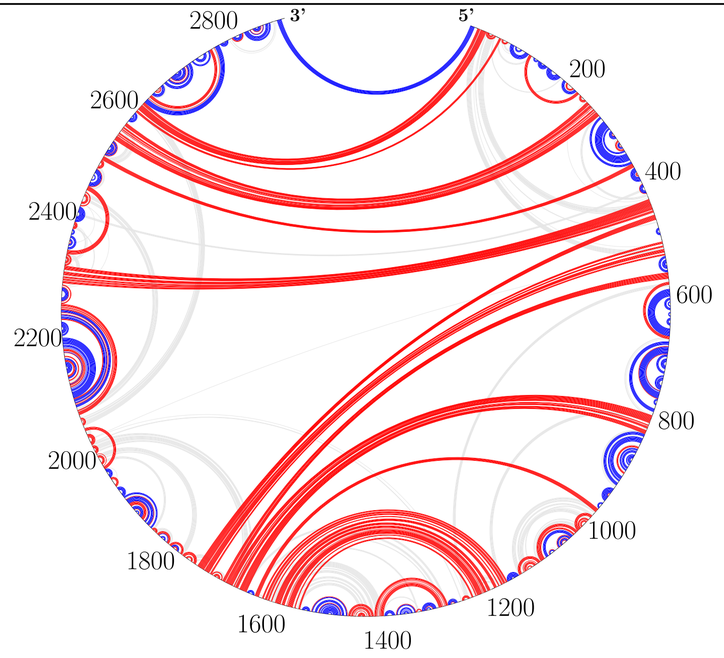
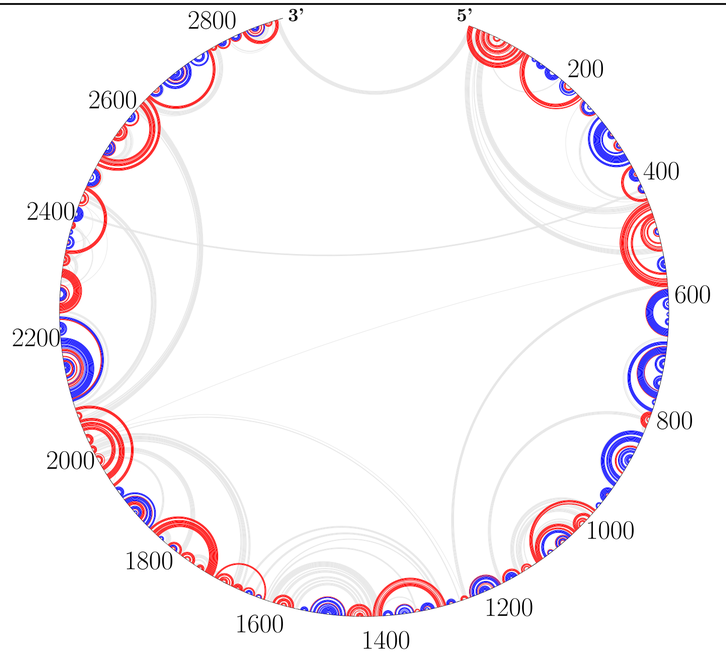
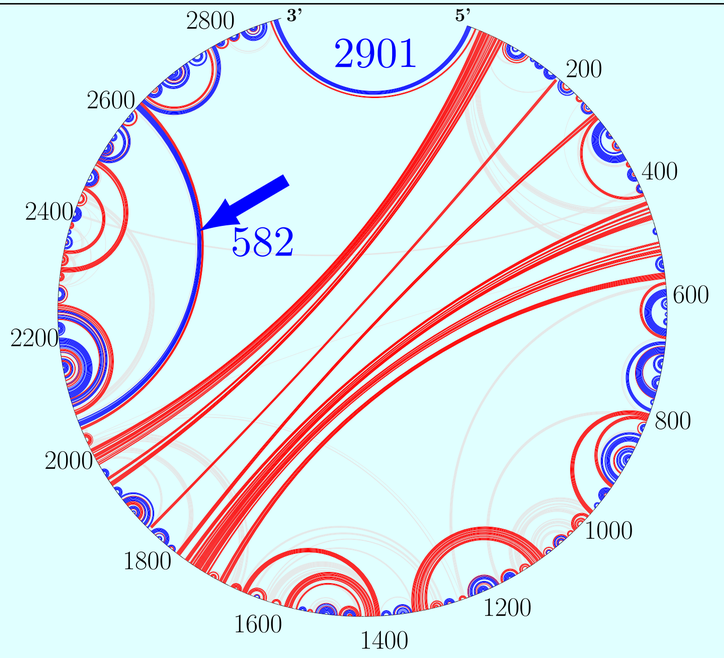
Baidu Research

UNIVERSITY of ROCHESTER

Backup Slide: Local Folding

type	*pair distance*	*time*	*space*	*systems/examples*
global	unbounded	$O(n^3)$	$O(n^2)$	RNAstructure, RNAfold, ...
local	$\leq L$	$O(nL^2)$	$O(nL)$	Rfold, RNAplfold, LocalFold, ...
global	unbounded	$O(nb \log b)$	$O(nb)$	LinearFold (**this work**)

23S rRNA *E. coli* (2904 *nt*, 830 pairs)

Vienna RNAfold	Vienna RNAfold Local (L=150)	LinearFold-V b = 100
		
PPV: 52.02, Sens.: 57.47, pairs: 917, time: 7.6s.	PPV: 51.63, Sens.: 55.18, pairs: 887, time: 6.3s.	PPV: 55.60, Sens.: 62.17, pairs: 928, time: 2.2s.