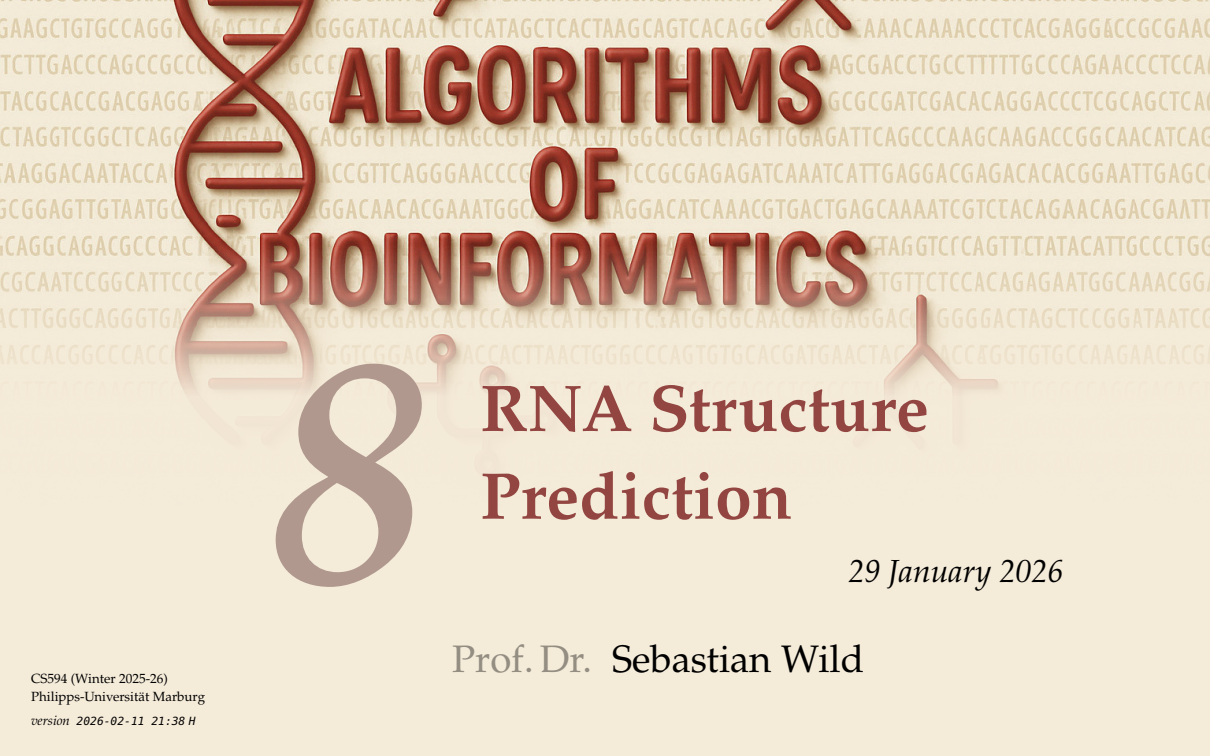


8 RNA Structure Prediction

29 January 2026

Prof. Dr. Sebastian Wild

CS594 (Winter 2025-26)
Philipps-Universität Marburg
version 2026-02-11 21:38 H

Outline

8 RNA Structure Prediction

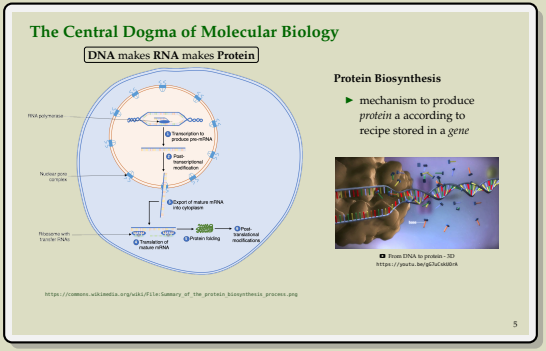
8.1 Noncoding RNA

RNA

RNA (Ribonucleic acid)

- similar to DNA: polymer of *nucleotides*
- ⇝ sequence of *nitrogenous bases* *Adenine*, and *Cytosine*, *Guanine*, *Uracil*
- unlike DNA, typically *single-stranded*
- more “sticky” backbone

- mostly known as *messenger RNA (mRNA)*
  - including *mRNA vaccines*!
  - mRNA is a coding RNA since they encode a protein



Noncoding RNA

But RNA serves many other roles!

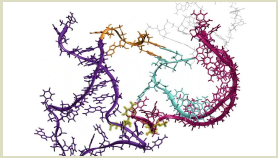
Introduction to Non-Coding RNA
https://youtu.be/KIohfQsRRdQ

- ironically, *ribosomes* (protein factories) themselves are mostly made of RNA
- for noncoding RNA, structure (3D folding form) crucial for function
- indeed, sequence often highly variable between species, but structure is similar!

RNA Secondary Structure Prediction

- Unfortunately, 3D shape hard and expensive to determine experimentally (*X-ray crystallography*)
- Available (diverse) data much smaller than for proteins
  - ⇝ May **not** soon see successful machine-learning solutions similar to AlphaFold

    Rhiju Das, `https://youtu.be/XqFq_zYx7Vo`
- To make matters worse, often not a single static structure



RNA folding in action
`https://youtu.be/2XTi9LG9NnU`

- ⇝ study *de-novo* approaches
- ⇝ and use simplified models of chemistry and shape to make progress

8.2 RNA Secondary Structure

Model of RNA Structure

- *RNA sequence / primary structure* $R[0..n) \in \Sigma^n$ $\quad \Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$
- *RNA secondary structure*: matching of indices $S \subset [0..n)^2$ of pairs (i, j) that are
  - ordered $i \leq j$
  - disjoint: $(i, j), (k, l) \in S \land (i = k \lor j = l) \implies (i, j) = (k, l)$
  - not too close $(i, j) \in S \implies j - i \geq 4$ backbone can't bend more
    min. length of hairpin loop
- secondary structure S is valid for sequence R if
  $(i, j) \in S \implies (R[i], R[j]) \in \mathcal{C} = \{(\mathtt{A}, \mathtt{U}), (\mathtt{U}, \mathtt{A}), (\mathtt{C}, \mathtt{G}), (\mathtt{G}, \mathtt{C}), (\mathtt{G}, \mathtt{U}), (\mathtt{U}, \mathtt{G})\}$
- $\mathcal{C}$ are the *canonical base pairs*: can form *hydrogen bonds* to stabilize RNA

Optimal RNA Structure – Attempt 1

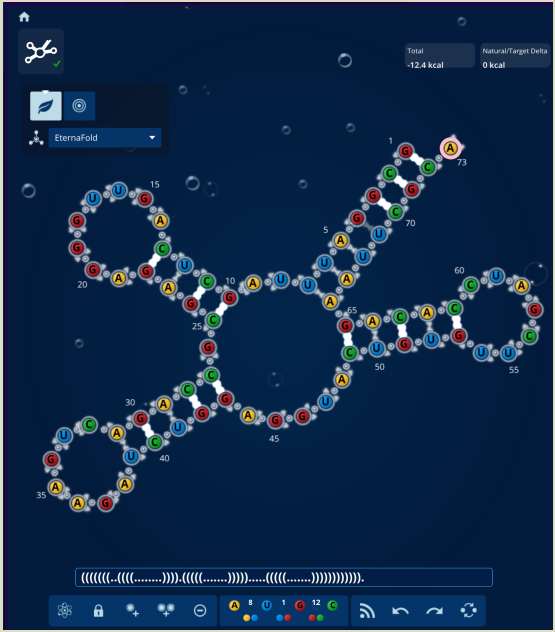
- Since base pairs provide stability
  Try to maximize $|S|$ (# pairs) among all valid secondary structures for $R[0..n)$.

$\rightsquigarrow$ **maximum matching** in graph of all bases

- possible in polynomial time
  - actually, ignoring minimum hairpin length, trivial greedy approach is optimal:
  1. form arbitrary `C` – `G` pairs (until we run out of `C`s or `G`s)
  2. form arbitrary `A` – `U` pairs (until we run out)
  3. form arbitrary `G` – `U` pairs (until we run out)

- unfortunately, useless predictions!
  - number of pairs dictated by base counts
  - many equally good options exist
  - many "optimal" solutions force entire molecule crowd up in one place

Let’s play a game!



phenylalanine transfer RNA from Saccharomyces
https://rnacentral.org/rna/URS000011107D/4930

EteRNA

`eternagame.org`

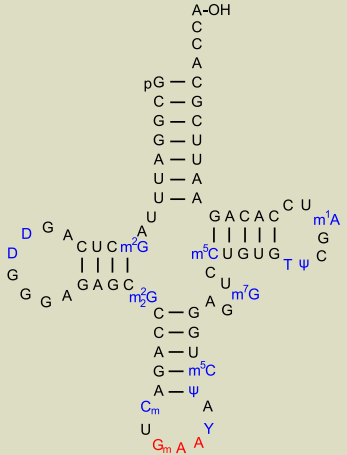
- Eterna is a citizen scientist computer game running since 2010
  lead by Rhiju Das (Stanford University School of Medicine)
- You have to design an RNA sequence that folds into a given *target secondary structure*.
- The game uses the best available simulation of RNA folding.
- Simulation, prediction, and RNA design algorithms are co-evolving
  - RNA design crowdsourced to players
  - top designs synthesized and structure determined

  ⇝ growing dataset for RNA structures

2D Approximation

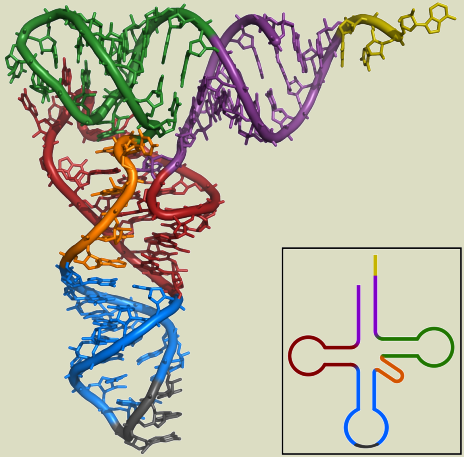
- As in Eterna, RNA secondary structure often drawn as *"roadkill diagrams"*

Roadkill diagram of yeast Phe tRNA



https://commons.wikimedia.org/wiki/File:TRNA-Phe_yeast_blanco.svg

3D Structure of yeast Phe tRNA



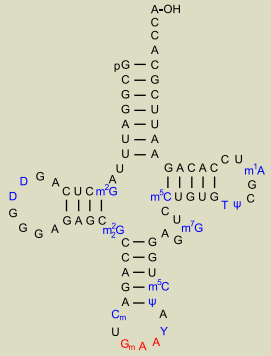
https://commons.wikimedia.org/wiki/File:TRNA-Phe_yeast_1ehz.png

Stacks

Key Observation: Stable structures have many **adjacent pairs**

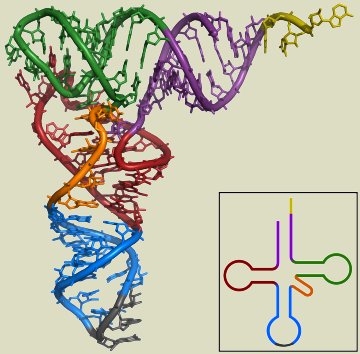
- *"stacked" pairs* forming a *stem* (the "ladder" regions)
- in 3D, stems form into a double helix (similar to DNA!)
- only reverse complement stems are stable

Roadkill diagram of yeast Phe tRNA



https://commons.wikimedia.org/wiki/File:TRNA-Phe_yeast_blanco.svg

3D Structure of yeast Phe tRNA

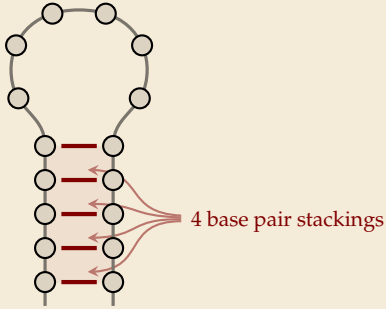


https://commons.wikimedia.org/wiki/File:TRNA-Phe_yeast_1ehz.png

Optimal RNA Structure – Attempt 2

- Recall: $S \subset [0..n)^2$ set of indices of paired bases
- instead of maximizing $|S|$ (# pairs), let's maximize number of **base pair stackings**!

$$\mathrm{BPS}(S) = \left|\left\{(i,j) \in S : (i+1, j-1) \in S\right\}\right|$$



General Secondary Structure Prediction

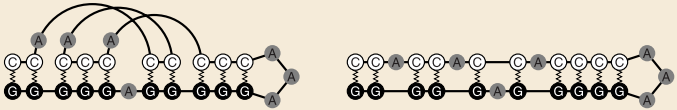
- **Given:** Sequence $R \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}^n$
- **Goal:** *Valid* secondary structure S with maximal $\mathrm{BPS}(S)$

Hardness

Unfortunately, General Secondary Structure Prediction is NP-hard.

- reduction from BinPacking

Lyngsø: *Complexity of Pseudoknot Prediction in Simple Models*, ICALP **2004**



(a) An optimum structure for the RNA sequence constructed from an instance of Bin Packing with four items of sizes 2, 2, 3, and 3, and two bins of capacity 5.

(b) An optimum structure for the RNA sequence constructed from an instance of Bin Packing with four items of sizes 2, 2, 2, and 4, and two bins of capacity 5.

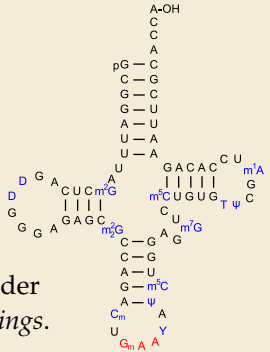
Fig. 3. Illustration of how the number of helices can be kept to one per item for an RNA sequence constructed from a 'yes' instance of Bin Packing, while the base pairs of at least one substring corresponding to an item have to be split over at least two helices if the RNA sequence is constructed from a 'no' instance of Bin Packing.

8.3 Pseudoknot-free secondary structures

Flat Structures

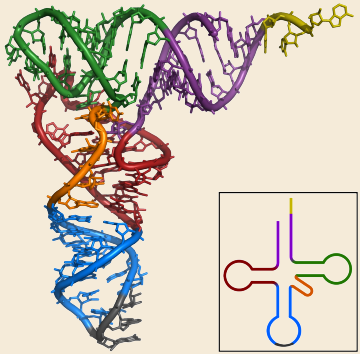
Recall example tRNA structure

Roadkill diagram of yeast Phe tRNA



https://commons.wikimedia.org/wiki/File:TRNA-Phe_yeast_blanco.svg

3D Structure of yeast Phe tRNA



https://commons.wikimedia.org/wiki/File:TRNA-Phe_yeast_1ehz.png

⇝ Seems reasonable to only consider roadkill diagrams *without crossings*.

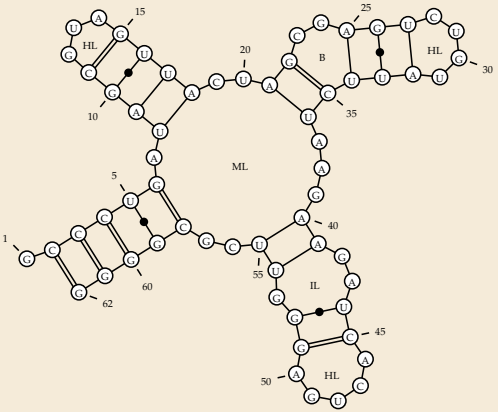
"Correct" formalization seems to be:
Require graph of pairs bases and backbone edges to be ***outerplanar***.

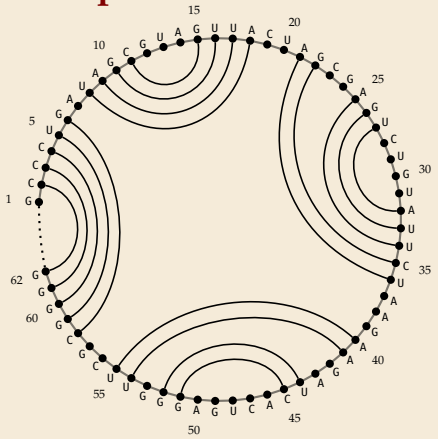
Any other secondary structure is called a *pseudoknot*.

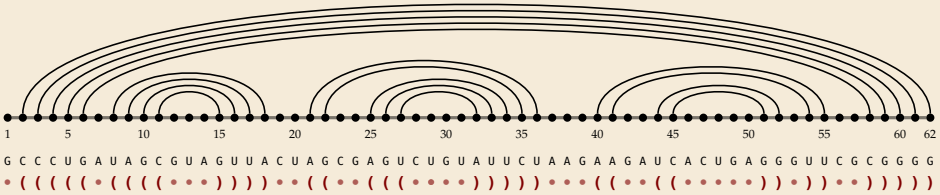
Pseudoknot-free secondary structures

- planar secondary structure (pairs) cover most of *free energy* of folding
- "coarse graining" of 3D structure biochemically useful
- natural intermediate step on folding pathway
- often well conserved between related species
- computationally tractable

Pseudoknot-free secondary structures – Representations







Nussinov's Algorithm

Idea: *Maximize total number of valid pairs among all* ***pseudoknot-free*** *structures.*

- back to maximum matching, but subject to outerplanar constraint . . .
- **outerplanar** *iff* $\forall (i,j),(k,\ell) \in S$ either
  $i < k < \ell < j$ (*nested* ), or
  $i < j < k < \ell$ (*disjoint* ).
  Anything else () $\rightsquigarrow$ *pseudoknot*.
- key insight: *decomposability!* see arc diagram / dot-bracket representation

$\rightsquigarrow$ Apply dynamic programming on subproblems $R[i..j)$

$D(i,j)$ = max valid pairs in pseudoknot-free structure for $R[i..j)$

Nussinov's Algorithm – DP

$D(i,j) =$ max valid pairs in any pseudoknot-free structure for $R[i..j)$

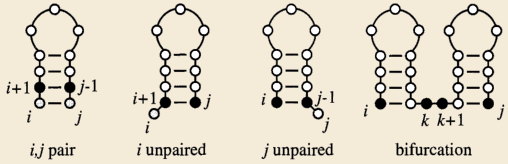


Figure 10.7 from Durbin et al. 1998

$$\rightsquigarrow \quad D(i,j) \;=\; \begin{cases} 0, & \text{if } j-i \le 4; \\ \max \begin{cases} D(i+1,j-1) + \big[(R[i],R[j{-}1]) \in \mathcal{C}\big], \\ D(i+1,j), \\ D(i,j-1), \\ \max\limits_{k \in [i..j)} D(i,k) + D(k+1,j) \end{cases} & \text{else.} \end{cases}$$

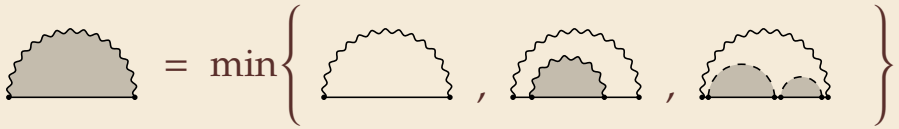
$\rightsquigarrow$ $O(n^3)$ time, $O(n^2)$ space

8.5 Refined Models

Back to Base Pair Stackings

- While maximum outerplanar matching is well-defined and tractable, it doesn't usually yield natural structures.
- already know that we should count base pair stackings!
- We can extend the DP solution to count those instead!

Graphical notation for DP recursions

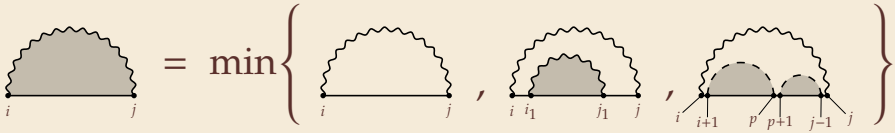


Key

- dots — bases; if touching, neighbors on backbone
- horizontal line — RNA backbone
- wiggly arcs — base pair
- dashed arcs — boundary; could be paired or not
- white area — no arcs here
- gray area — potentially further arcs

Counting Base Pair Stackings

Idea: Need to remember whether outermost bases paired.



- In the middle case, if $(i_1, j_1) = (i, j)$, count stacked base pair for (i, j)

$\rightsquigarrow$ Same $O(n^3)$ time, $O(n^2)$ space complexity

Turner Energy Model

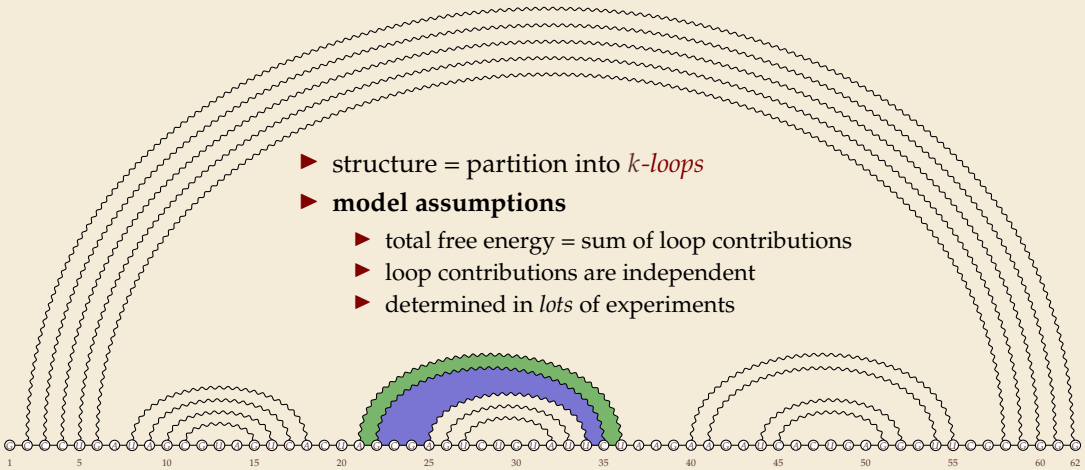
- Simply counting base pair stackings is still a **very crude approximation**
- Which bases are paired influences bonding strength
- Which bases are adjacent in stems influences stabilization contribution of stem
- Which bases form first unpaired base in hairpin loop influences stability
- . . . (play Eterna a bit for more 😉)

⇝ More refined models to compute free energy (≈ instability) of structure

Mathews, Sabina, Zuker, Turner: *Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure,* J Molecul. Biolog. **1999**

Mathews, Disney, Childs, Schroeder, Zuker, Turner: *Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure,* PNAS **2004**

Turner Energy Model [2]



- structure = partition into *k-loops*
- **model assumptions**
  - total free energy = sum of loop contributions
  - loop contributions are independent
  - determined in *lots* of experiments

Conceptually unbounded sum

too many variables!

Zuker's Algorithm

- Only compute exactly up to 2-loops (2 enclosed pairs)
- additive approximation for bigger multiloops

$\rightsquigarrow$ *same* mutually recursive cost as for pair stackings

8.6 Grammar-based Approaches

Can’t machine learning help?

- free-energy models are great *ab initio* methods
- however, they remain limited in accuracy
- with growing datasets, tempting to improve structure prediction using machine learning

- but: available data much too few for blackbox learning

⇝ statistical learning with curated probabilistic model

Probabilistic Context-Free Grammars

Recap from your formal languages intro course . . .

Context-free grammars (CFG)

$G = (N, T, R, S)$

- nonterminals N
- terminals T
- rules $R \subseteq N \times (N \cup T)^*$
- start symbol $S \in N$

Applying rules to replace nonterminals

$S \Rightarrow^* w \quad \rightsquigarrow \quad w \in \mathcal{L}(G)$

Example

- $N = \{E, I, V, C, C'\}$
- $T = \{\texttt{x}, \texttt{y}, \texttt{0}, \ldots, \texttt{9}, \texttt{+}, \bullet\}$
- $E \to (E + E) \mid (E \bullet E) \mid I$
  $I \to C \mid V$
  $V \to \texttt{x} \mid \texttt{y}$
  $C \to \texttt{0} \mid \texttt{1}C' \mid \ldots \mid \texttt{9}C'$
  $C' \to \varepsilon \mid \texttt{0}C' \mid \ldots \mid \texttt{9}C'$

  ε: empty string

Probabilistic Context-Free Grammars (PCFG)

- For each nonterminal, assign *probabilities* to right-hand sides.

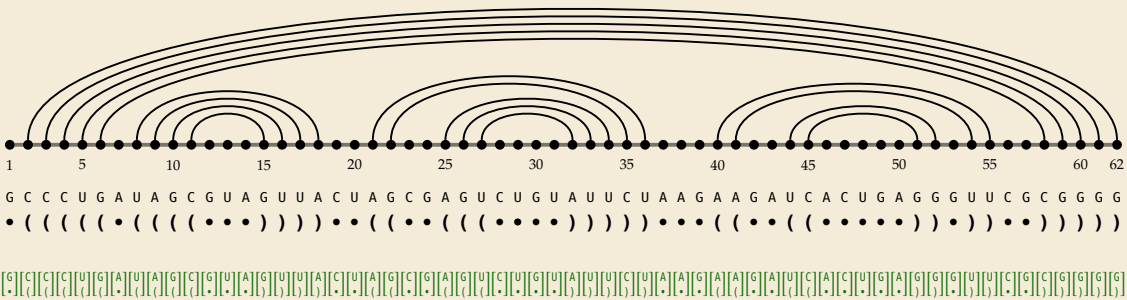
$\rightsquigarrow$ prob of a derivation in G = product of rule probabilities.

RNA Sequence and Structure

Pseudoknot-free secondary structures $\approx$ correctly parenthesized strings

$\rightsquigarrow$ representable by context-free grammar

Idea: Represent RNA sequence and structure as string of pairs



$\rightsquigarrow$ Consider grammars with terminals

$$T = \left\{ \left[\begin{smallmatrix}A\\\cdot\end{smallmatrix}\right], \left[\begin{smallmatrix}A\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}A\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}C\\\cdot\end{smallmatrix}\right], \left[\begin{smallmatrix}C\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}C\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}G\\\cdot\end{smallmatrix}\right], \left[\begin{smallmatrix}G\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}G\\)\end{smallmatrix}\right], \left[\begin{smallmatrix}U\\\cdot\end{smallmatrix}\right] \left[\begin{smallmatrix}U\\(\end{smallmatrix}\right], \left[\begin{smallmatrix}U\\)\end{smallmatrix}\right] \right\}$$

RNA Grammars

Example Grammar to produce RNA sequence and structure pairs

(with some fictitious rule probabilities)

rule	prob.	rule	prob.	rule	prob.
$S \to L\,S$	0.65	$L \to \begin{bmatrix}\mathtt{C}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.10	$L \to \begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}$	0.10
$S \to \varepsilon$	0.35	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{C}\\)\end{bmatrix}$	0.05	$L \to \begin{bmatrix}\mathtt{U}\\\bullet\end{bmatrix}$	0.15
$L \to \begin{bmatrix}\mathtt{A}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.05	$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{G}\\)\end{bmatrix}$	0.05	$L \to \begin{bmatrix}\mathtt{C}\\\bullet\end{bmatrix}$	0.10
$L \to \begin{bmatrix}\mathtt{U}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{A}\\)\end{bmatrix}$	0.15	$L \to \begin{bmatrix}\mathtt{G}\\(\end{bmatrix} S \begin{bmatrix}\mathtt{U}\\)\end{bmatrix}$	0.10	$L \to \begin{bmatrix}\mathtt{G}\\\bullet\end{bmatrix}$	0.15

Can prove: Grammar has unique parse tree *iff* valid RNA sequence structure pair.

Key idea for structure prediction:

- Given only sequence of bases, cannot distinguish $\begin{bmatrix}\mathtt{A}\\\bullet\end{bmatrix}$, $\begin{bmatrix}\mathtt{A}\\(\end{bmatrix}$, and $\begin{bmatrix}\mathtt{A}\\)\end{bmatrix}$
- $\rightsquigarrow$ Many possible parse trees compatible with sequence
- $\rightsquigarrow$ Predict structure corresponding to *most likely parse tree*

Structure Prediction with PCFG

rule	prob.	rule	prob.	rule	prob.
$S \to L\,S$	0.65	$L \to \left[\begin{smallmatrix}\mathtt{C}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\mathtt{G}\\)\end{smallmatrix}\right]$	0.10	$L \to \left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]$	0.10
$S \to \varepsilon$	0.35	$L \to \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\mathtt{C}\\)\end{smallmatrix}\right]$	0.05	$L \to \left[\begin{smallmatrix}\mathtt{U}\\\cdot\end{smallmatrix}\right]$	0.15
$L \to \left[\begin{smallmatrix}\mathtt{A}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\mathtt{U}\\)\end{smallmatrix}\right]$	0.05	$L \to \left[\begin{smallmatrix}\mathtt{U}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\mathtt{G}\\)\end{smallmatrix}\right]$	0.05	$L \to \left[\begin{smallmatrix}\mathtt{C}\\\cdot\end{smallmatrix}\right]$	0.10
$L \to \left[\begin{smallmatrix}\mathtt{U}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\mathtt{A}\\)\end{smallmatrix}\right]$	0.15	$L \to \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right] S \left[\begin{smallmatrix}\mathtt{U}\\)\end{smallmatrix}\right]$	0.10	$L \to \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]$	0.15

Leftmost derivations for $R = \mathtt{GAC}$ using $\mathtt{A} \equiv \left[\begin{smallmatrix}\mathtt{A}\\*\end{smallmatrix}\right] \equiv \left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right] \equiv \left[\begin{smallmatrix}\mathtt{A}\\(\end{smallmatrix}\right] \equiv \left[\begin{smallmatrix}\mathtt{A}\\)\end{smallmatrix}\right]$

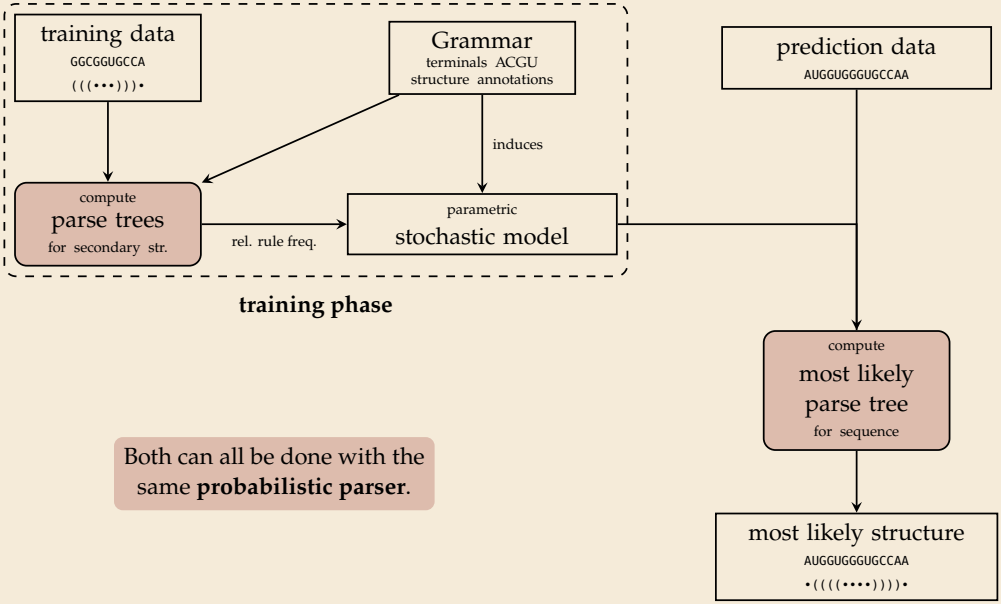
1. $\dot{S} \underset{0.65}{\Rightarrow} \dot{L}S \underset{0.15}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]\dot{S} \underset{0.65}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]\dot{L}S \underset{0.10}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\dot{S} \underset{0.65}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\dot{L}S \underset{0.10}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{C}\\\cdot\end{smallmatrix}\right]\dot{S} \underset{0.35}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{C}\\\cdot\end{smallmatrix}\right]$
   Total probability 0.000144

2. $\dot{S} \underset{0.65}{\Rightarrow} \dot{L}S \underset{0.05}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right]\dot{S}\left[\begin{smallmatrix}\mathtt{C}\\)\end{smallmatrix}\right]S \underset{0.65}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right]\dot{L}S\left[\begin{smallmatrix}\mathtt{C}\\)\end{smallmatrix}\right]S \underset{0.10}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\dot{S}\left[\begin{smallmatrix}\mathtt{C}\\)\end{smallmatrix}\right]S \underset{0.35}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{C}\\)\end{smallmatrix}\right]\dot{S} \underset{0.35}{\Rightarrow} \left[\begin{smallmatrix}\mathtt{G}\\(\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{A}\\\cdot\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathtt{C}\\)\end{smallmatrix}\right]$
   Total probability 0.000259

$\rightsquigarrow$ Predict structure "`(•)`" (Simple grammar ignores min length of hairpin loops!)

Here trivial, since only 2 different derivations . . . how to solve in general?
And where do the probabilities come from?

Stochastic Approach to Secondary Structure Prediction



Both can all be done with the same **probabilistic parser**.

8.7 Probabilistic Parsing

Refresher: Chomsky Normal Form

Describing parsing easier with specific form of grammars. *(reduces case distinctions)*

Theorem 8.1 (Chomsky-Normalform (CNF))

For every context-free grammar (CFG) $G = (N, T, R, S)$ with $\varepsilon \notin \mathcal{L}(G)$, we can effectively construct CFG $G' = (N', T, R', S')$ with $\mathcal{L}(G) = \mathcal{L}(G')$ and $R' \subset N' \times N'N' \cup N' \times T$. ◀

$\rightsquigarrow$ Only rules of form $A \to a$ or $A \to BC$

Proof idea:

Successively eliminate

1. ε-rules $A \to \varepsilon$
2. chain rules $A \to B$
3. "unproductive nonterminals"
   i. e., A where there is *no* derivation $S \Rightarrow^\star \alpha A \beta \Rightarrow^\star w$
4. terminals in RHS $A \to \alpha \mathtt{a} \beta \quad \rightsquigarrow \quad A \to \alpha A_{\mathtt{a}} \beta$ und $A_{\mathtt{a}} \to a$
5. overlong RHS $A \to BCD \quad \rightsquigarrow \quad A \to BA'$ and $A' \to CD$

$\rightsquigarrow$ need to verify:

Later steps don't re-introduce problems eliminated earlier

(easy)

Refresher: Cocke-Younger-Kasami

CNF $\rightsquigarrow$ Only rules of form $A \to a$ or $A \to BC$

Given: CFG $G = (N, T, R, S)$ in Chomsky normal form, word $w \in T^n$, $n \geq 1$
Decide: $w \in \mathcal{L}(G)$?

Solve **more general** problem: $A \Rightarrow^\star w[i..j)$**?** for **all** $A \in N, 0 \leq i < j \leq n$.

$\rightsquigarrow$ $w \in \mathcal{L}(G)$ *iff* $A \Rightarrow^\star w[i..j)$ for $A = S$, $i = 0$, $j = n$

- Guess **first used rule** $\rightsquigarrow$ two cases for the two rule types
- $A \Rightarrow^\star w[i..i+1)$ *iff* $A \to w[i] \in R$
- $A \Rightarrow^\star w[i..j)$ for $j \geq i+2$ *iff* $\exists k \in [i+1..j), B, C \in N$ with $A \to BC \in R$ and

  instances of same problem: $B \Rightarrow^\star w[i..k)$ and $C \Rightarrow^\star w[k..j)$

$\rightsquigarrow$ **dynamic programming** $\rightsquigarrow$ $O(n^3 g)$ time for g the number of rules in G

$\rightsquigarrow$ via backtrace also find a **leftmost derivation** for w (if $w \in \mathcal{L}(G)$)

Probabilistic Parsing

Notation

- derivation $r_1, \ldots, r_t$ = a sequence of rules from R
- probability of a derivation $\mathbb{P}[r_1, \ldots, r_t] = \prod_{i=1}^{t} P(r_i)$
- write $\mathrm{lmd}_{A \to \gamma}(xA\beta) = x\gamma\beta$ for one leftmost derivation step
- extend $\mathrm{lmd}_r(\circ)$ to several rules $\mathrm{lmd}_{r_1,\ldots,r_t}(\circ)$ by successively applying rules

Standard CYK solves the **word/parsing problem**

Given: CFG $G = (N, T, R, S)$ in Chomsky normal form, word $w \in T^n$, $n \geq 1$
Output: any $r_1, \ldots, r_t$ such that $\mathrm{lmd}_{r_1,\ldots,r_t}(S) = w$ or `NOT_IN_LANGUAGE`

For RNA structure prediction, need the most likely among many possible derivations!

Viterbi parse problem

Given: PCFG $G = (N, T, R, S, P)$, word $w \in T^n$, $n \geq 1$
Output: $r_1, \ldots, r_t = \underset{\mathrm{lmd}_{r_1,\ldots,r_t}(S)=w}{\arg\max_{r_1,\ldots,r_t}} \mathbb{P}[r_1, \ldots, r_t]$

Viterbi value vs. probability

Note that in probabilistic parsing, there two "probabilities" for a string w

1. the *Viterbi value*

   $$V(w) \;=\; \max_{\substack{r_1,\ldots,r_t \\ \mathrm{lmd}_{r_1,\ldots,r_t}(S)=w}} \mathbb{P}[r_1,\ldots,r_t]$$

   probability of a single, most likely derivation $S \Rightarrow^\star w$

2. the *production probability*

   $$\mathbb{P}[w] \;=\; \mathbb{P}[S \Rightarrow^\star w] \;=\; \sum_{\substack{r_1,\ldots,r_t \\ \mathrm{lmd}_{r_1,\ldots,r_t}(S)=w}} \mathbb{P}[r_1,\ldots,r_t]$$

   total probability that we obtain w
   when starting with S and applying randomly chosen rules to expand nonterminals.

Both can make sense; for us only Viterbi value needed.

Probabilistic CYK

In Chomsky normal form, easy to make CYK probabilistic

General Viterbi values: $V_A(w) = \max_{\substack{r_1,\ldots,r_t \\ \mathrm{lmd}_{r_1,\ldots,r_t}(A)=w}} \mathbb{P}[r_1,\ldots,r_t]$

$$V_A(w[i..i+1)) = \begin{cases} P(A \to w[i]) & \text{if } A \to w[i] \in R \\ 0 & \text{otherwise} \end{cases}$$

$$V_A(w[i..j)) = \max_{\substack{k\in[i+1..j) \\ A\to BC\in R}} P(A \to BC)\, V_B(w[i..k))\, V_C(w[k..j))$$

$\rightsquigarrow$ Same DP works!
$O(n^3 g)$ time, backtrace yields most likely derivation

8.8 The Grammar of RNA

RNA Grammar Normal Form

Chomsky Normal Form rather inconvenient for RNA.

- Really want a single rule to introduce a pair bond.
- Nonterminals often represent a logical abstraction of several special cases would want to have chain rules for that

Stochastic RNA Form (SRF)

Normal form from **Onokpasa, Wild, Wong**: *Towards Optimal Grammars for RNA Structures*, DCC **2024**

- Assume $N = \{A_1, \ldots, A_k\}$
- *start symbol* must be A_k
- all rules must of the form
  - (i) $A_i \to A_j A_\ell$
  - (ii) $A_i \to \bullet$
  - (iii) $A_i \to (A_j)$
  - (iv) $A_i \to A_j$ and $j < i$

  (assumes a *total order* on the nonterminals)

Viterbi values

Generalization of CYK computes Viterbi values

$$V_A(w[i..i+1)) \;=\; \begin{cases} P(A \to w[i]) & \text{if } A \to w[i] \in R \quad \text{type (ii) rules} \\ 0 & \text{otherwise} \end{cases}$$

$$V_A(w[i..j)) \;=\; \max \begin{cases} \max\limits_{\substack{k \in [i+1..j) \\ A \to BC \in R}} P(A \to BC)\, V_B(w[i..k))\, V_C(w[k..j)) & \text{type (i) rules} \\ \max\limits_{A \to w[i] B\, w[j-1] \in R} P(A \to w[i]\, B\, w[j-1])\, V_B(w[i+1..j-1)) & \text{type (iii) rules} \\ \max\limits_{A \to B \in R} P(A \to B)\, V_B(w[i..j)) & \text{type (iv) rules} \end{cases}$$

Towards Subcubic RNA Folding

- both general context-free parsing as well as minimum-free energy models led to $\Theta(n^3)$ worst case time algorithms
- at core, both problems resemble matrix multiplication
  - dominant running time comes from bifurcation rules
  - e. g., Recall Nussinov $D(i,j) = \begin{cases} 0, & \text{if } j - i \leq 4; \\ \max \begin{cases} D(i+1, j-1) + \left[(R[i], R[j-1]) \in \mathcal{C}\right], \\ D(i+1, j), \\ D(i, j-1), \\ \max\limits_{k \in [i..j)} D(i,k) + D(k+1, j) \end{cases} & \text{else.} \end{cases}$

Towards Subcubic RNA Folding [2]

Approaches

1. Exhaustive Tabulation can be made to work . . . shaves off $\log$-factors
2. for (probabilistic) context-free parsing *Valiant's algorithm* reduces the problem to (Boolean) matrix multiplication $\rightsquigarrow$ $O(n^\omega)$ possible

   Valiant: *General context-free recognition in less than cubic time*, J Comput. System Sci. **1975**

   - constant factors are huge; even asymptotically slower Strassen
   - beyond plain word recognition problem, more complications to be overcome

     Zakov, Tsur, Ziv-Ukelson: *Reducing the worst case running times of a family of RNA and CFG problems, using Valiant's approach*, Alg. Molecul. Biol. **2011**

3. Closer to $(\min, +)$-matrix multiplication than regular matrix $(+, \cdot)$-multiplication
   - in general, likely no $O(n^{3-\epsilon})$ algorithms to be expected
   - but for bounded difference matrices, speedups known

     Bringmann, Grandoni, Saha, Williams: *Truly Subcubic Algorithms for Language Edit Distance and RNA Folding via Fast Bounded-Difference Min-Plus Product*, SIAM J. Comput. **2019**

What Grammars to Choose?

- #parameters = #rules −1 per nonterminal
- more parameters means more expressive
  but also more prone to overfitting
- finding a grammar that captures structure well at this point *more art than science*

⇝ Good results, e. g., for this grammar

$$
\begin{aligned}
U &\to \bullet \\
B &\to (M) \\
T &\to B \mid U \\
M &\to B \mid TS \mid T \\
S &\to TS \mid T
\end{aligned}
$$

Grammar G_6 from

Dowell, Eddy: *Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction*, BMC Bioinform. **2004**

8.9 RNA Prediction and Compression

Prediction = Compression + Generalization

- RNA grammar turns RNA sequence-structure pair into single string: The sequence of used rules!
- Can compress these rules using arithmetic coding

⇝ efficient compression of RNA

Can we use the compression ratio as a proxy to compare RNA grammars?

Onokpasa, Wild, Wong: *RNA secondary structures: from ab initio prediction to better compression, and back,* DCC **2023**

Can we find better grammars than Dowell & Eddy?

Onokpasa, Wild, Wong: *Towards Optimal Grammars for RNA Structures,* DCC **2024**