

CS481/CS583: Bioinformatics Algorithms

Can Alkan




EA509

calkan@cs.bilkent.edu.tr

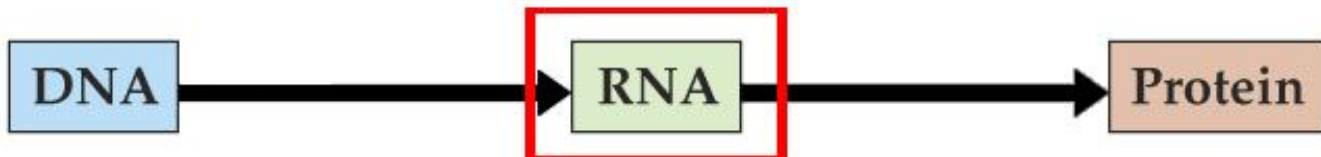
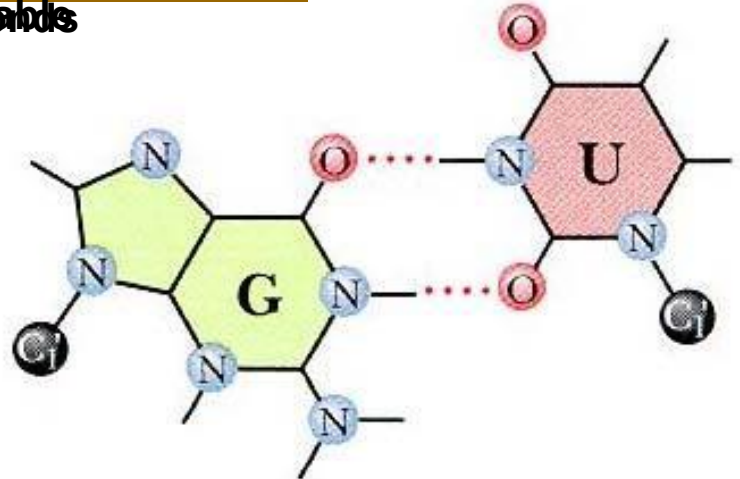
<http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/>

RNA STRUCTURE

RNA Basics

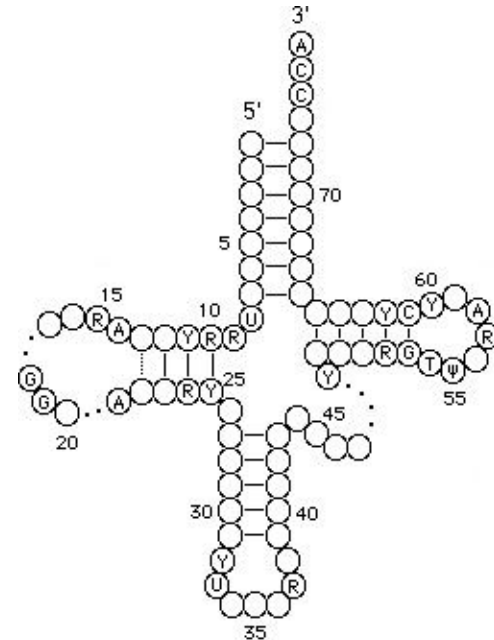
- RNA bases A,C,G,U
- Canonical Base Pairs
 - ❑ A-U 
 - ❑ G-C 
 - ❑ G-U 
“wobble” pairing
- ❑ Bases can only pair with **one** other base.

3 Hydrogen Bonds – more
Stable



RNA Basics

- transfer RNA (tRNA)
- messenger RNA (mRNA)
- ribosomal RNA (rRNA)
- small interfering RNA (siRNA)
- micro RNA (miRNA)
- small nucleolar RNA (snoRNA)
- Long non-coding RNA (lncRNA)
-

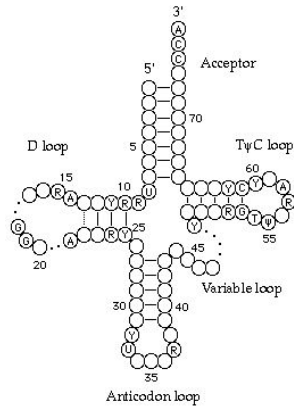


RNA folding

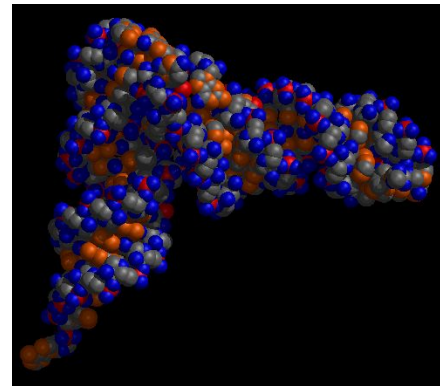
- Prediction of secondary structure of an RNA given its sequence
 - General problem is NP-hard due to “difficult” substructures, like pseudoknots
 - Most existing algorithms require too much memory ($\geq O(n^2)$), and run time ($\geq O(n^3)$) thus limited to smaller RNA sequences
-

RNA Structural Levels

AAUCG...CUUCUUCCA
Primary



Secondary



Tertiary

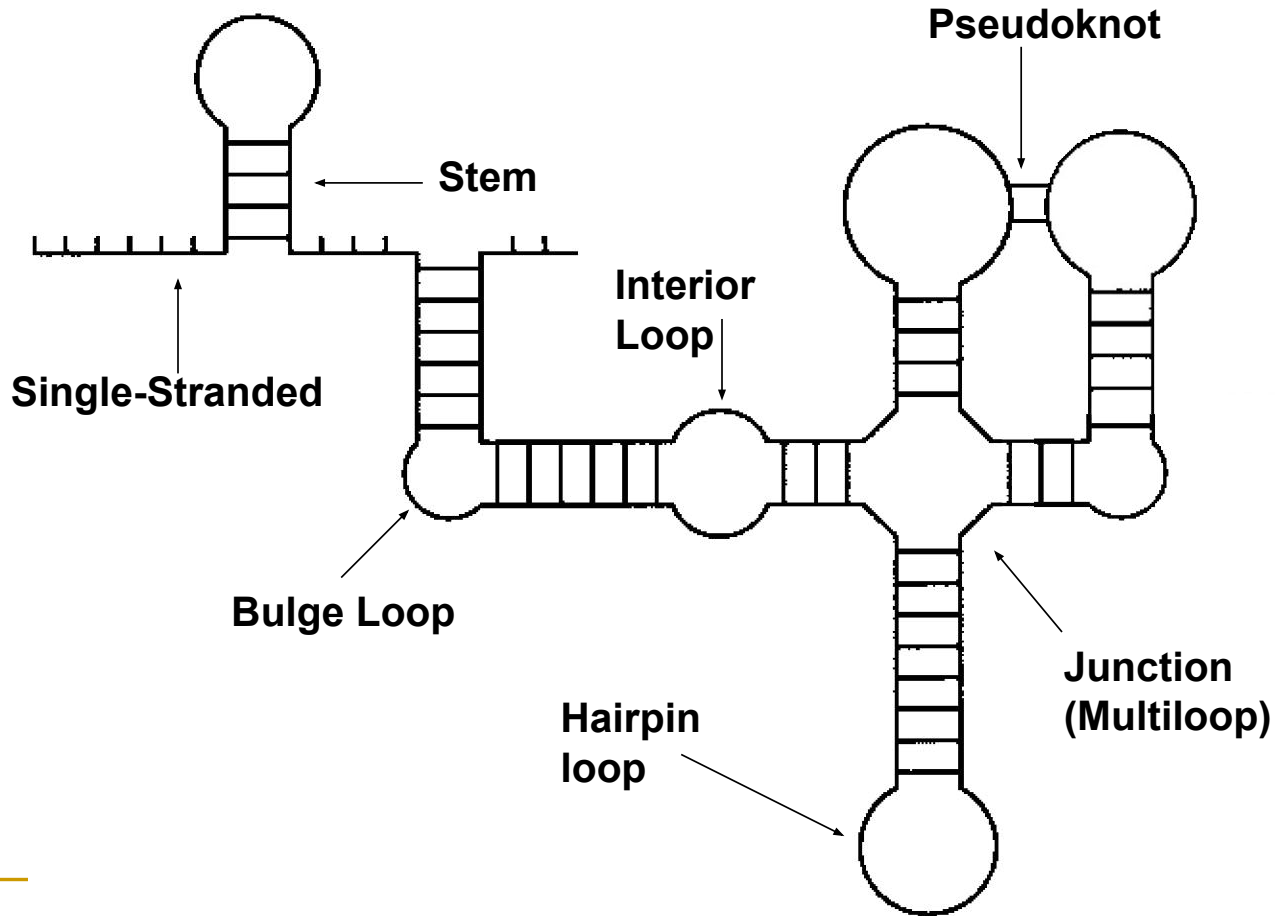
RNA families

- Rfam : General non-coding RNA database
(most of the data is taken from specific databases)

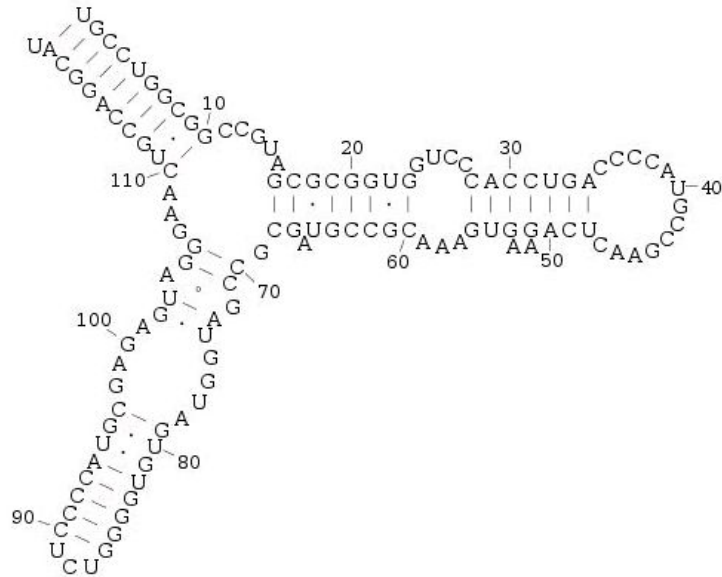
<http://www.sanger.ac.uk/Software/Rfam/>

Includes many families of non coding RNAs and functional Motifs, as well as their alignment and their secondary structures

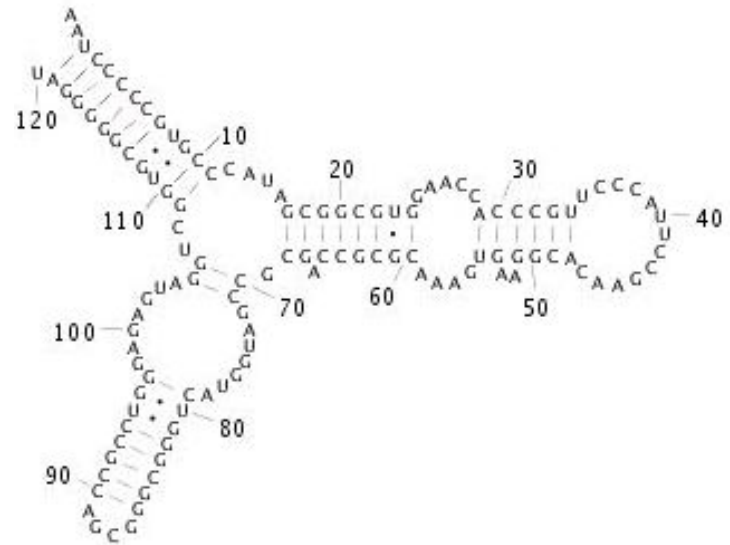
RNA Secondary Structure



Example: 5S rRNA

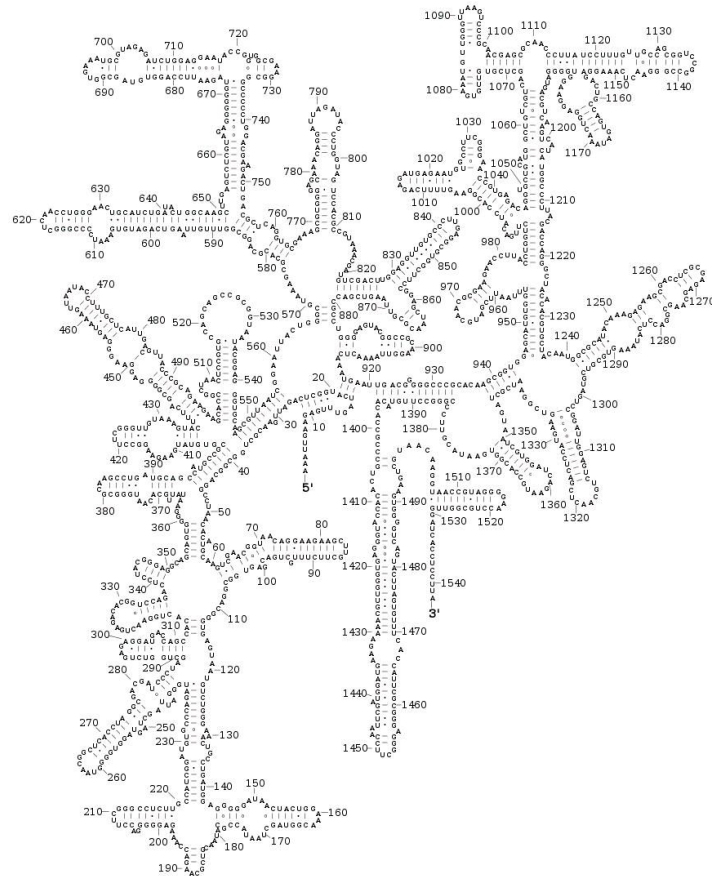


E. coli 5S
120 bases



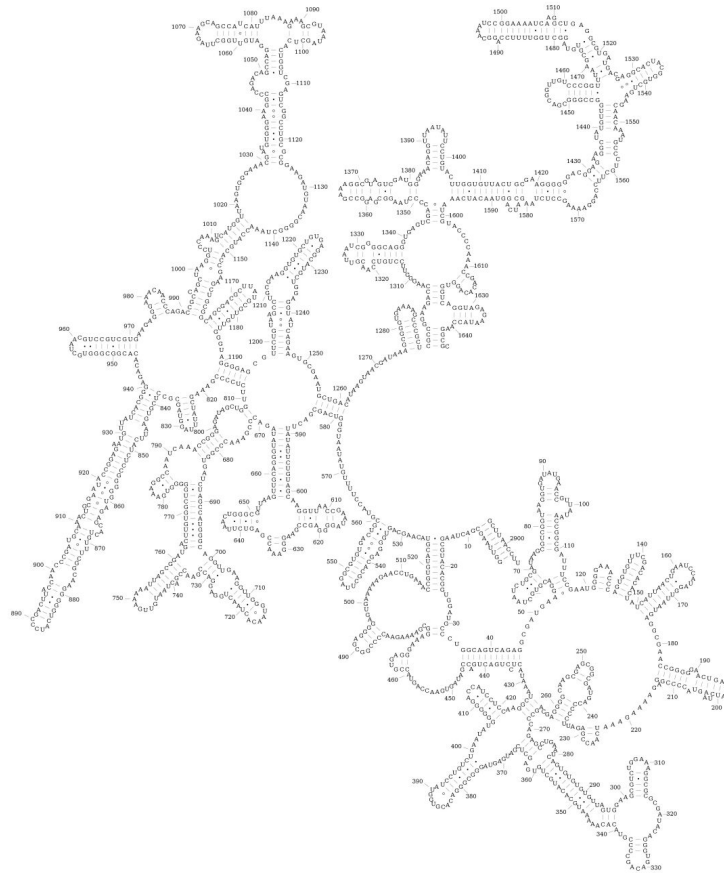
T. thermophilus 5S
120 bases

Example: E. coli 16S rRNA



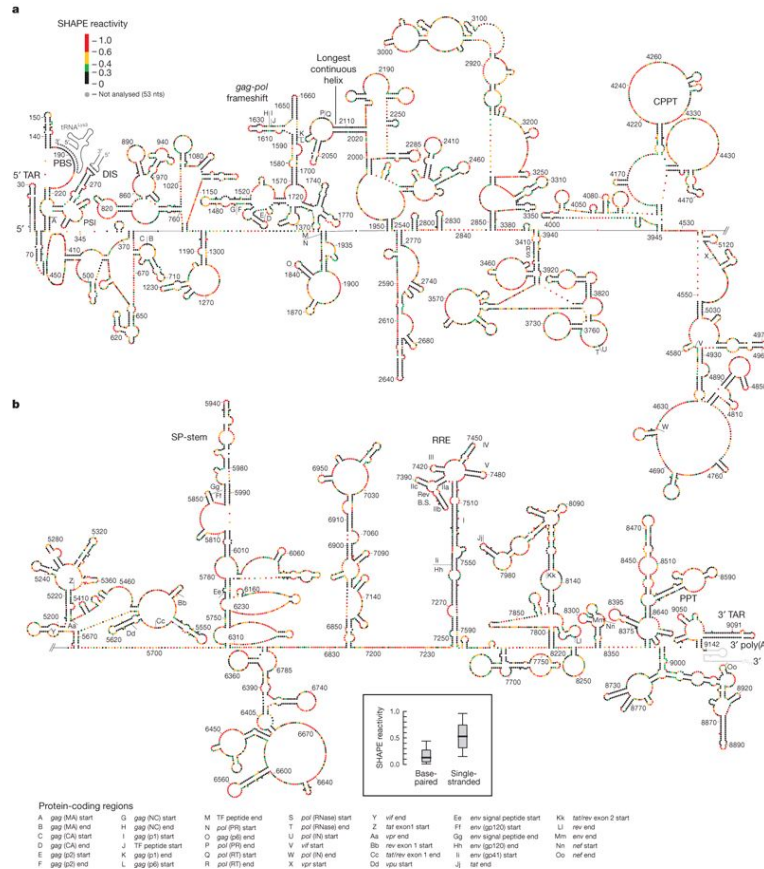
1542 bases

Example: *E. coli* 23S rRNA



2904 bases

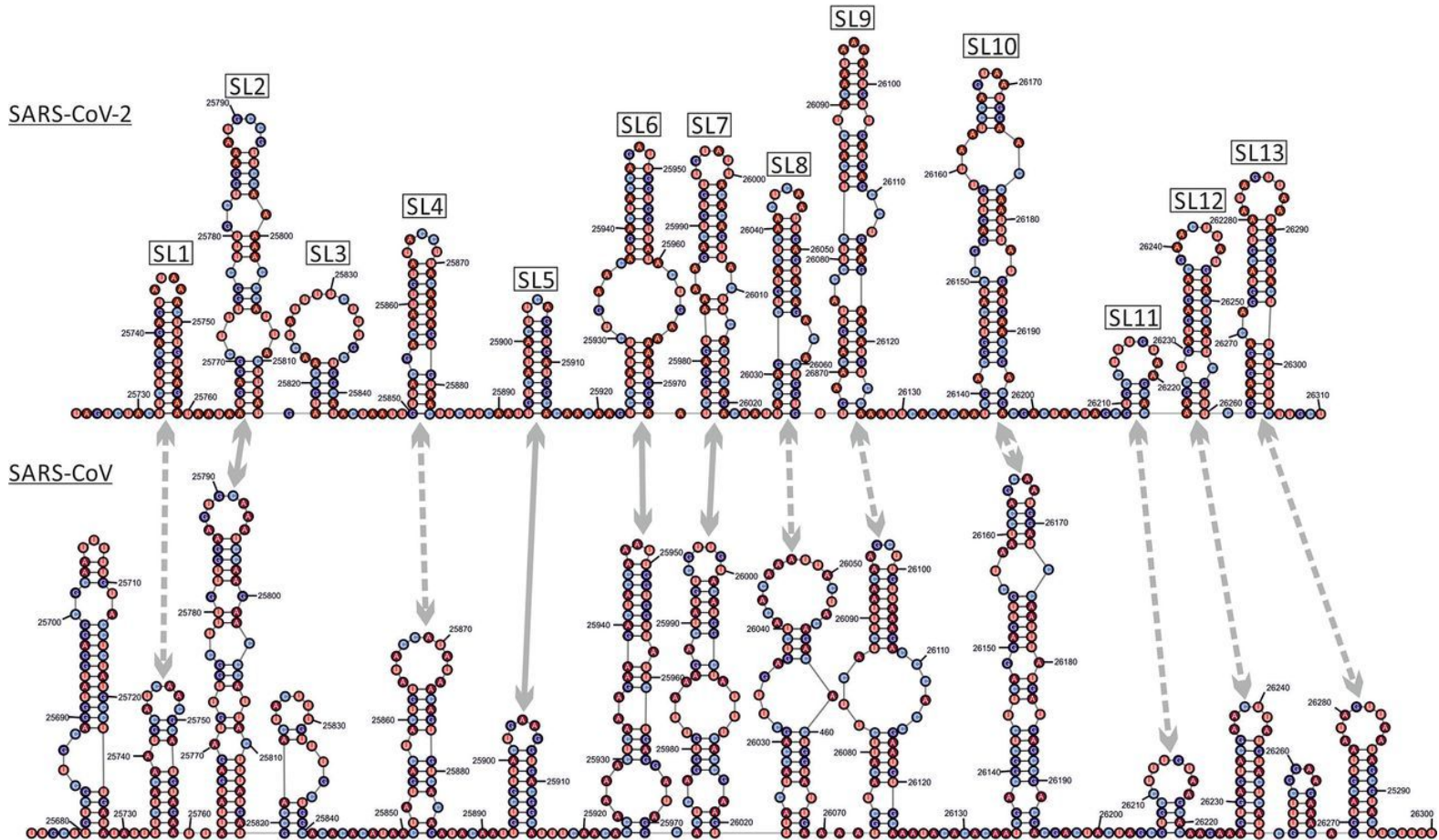
Example: HIV



9173 bases

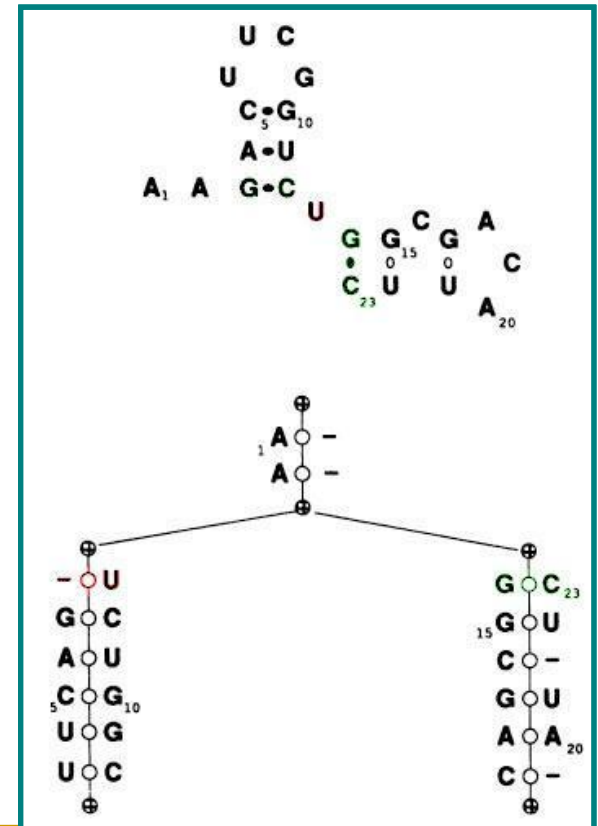
Watts et al., Nature, 2009

Example: SARS-CoV and SARS-CoV-2



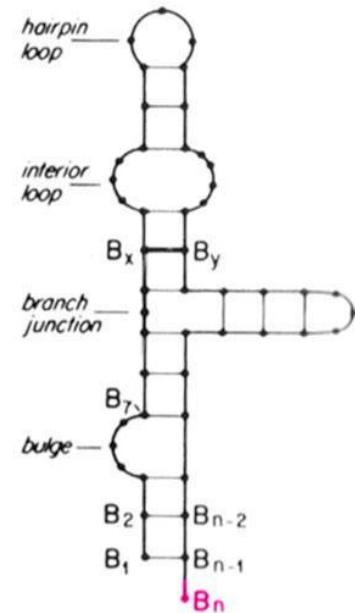
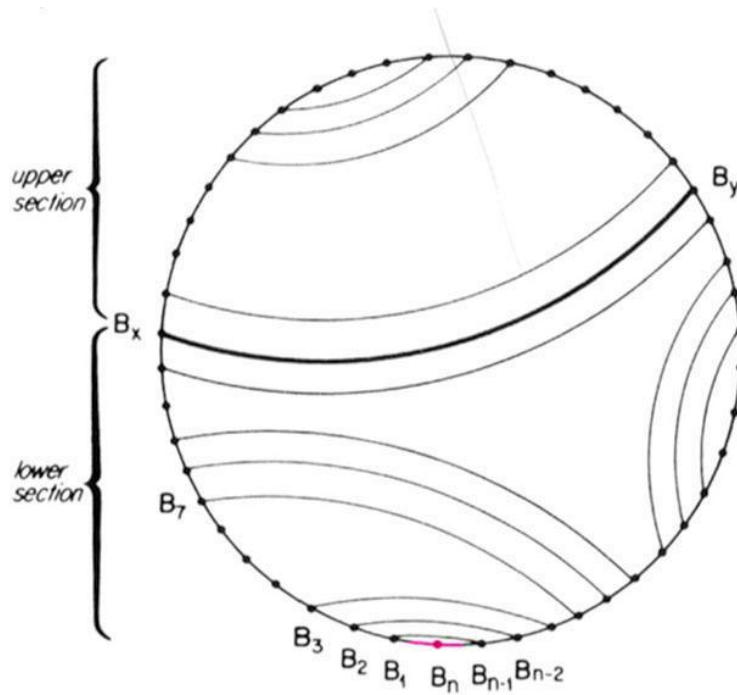
Binary Tree Representation of RNA Secondary Structure

- Representation of RNA structure using Binary tree
- Nodes represent
 - Base pair if two bases are shown
 - Loop if base and “gap” (dash) are shown
- Pseudoknots still not represented
- Tree does not permit varying sequences
 - Mismatches
 - Insertions & Deletions



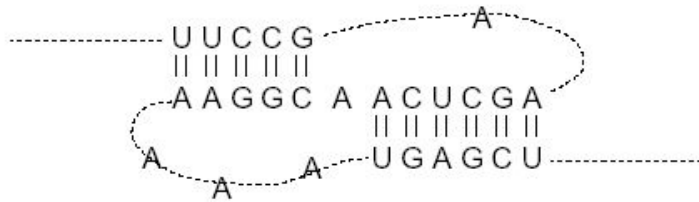
Images – Eddy et al.

Circular Representation

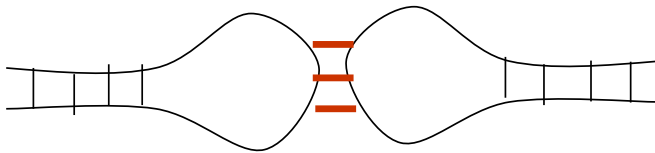


Examples of known interactions of RNA secondary structural elements

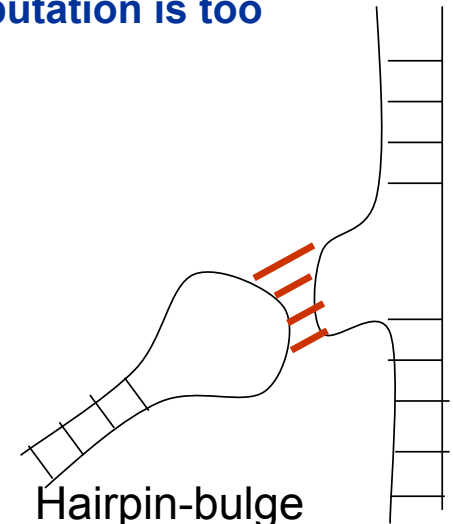
Pseudoknot



These patterns are excluded from the prediction schemes as their computation is too intensive.



Kissing hairpins



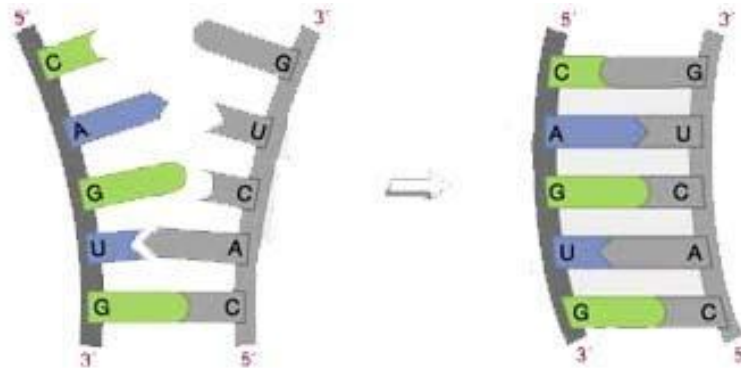
Hairpin-bulge contact

Predicting RNA secondary structure

- Base pair maximization
- Minimum free energy (most common)
 - Fold, Mfold (Zuker & Stiegler)
 - RNAfold (Hofacker)
- Multiple sequence alignment
 - Use known structure of RNA with similar sequence
- Covariance
- Stochastic Context-Free Grammars

Sequence Alignment as a method to determine structure

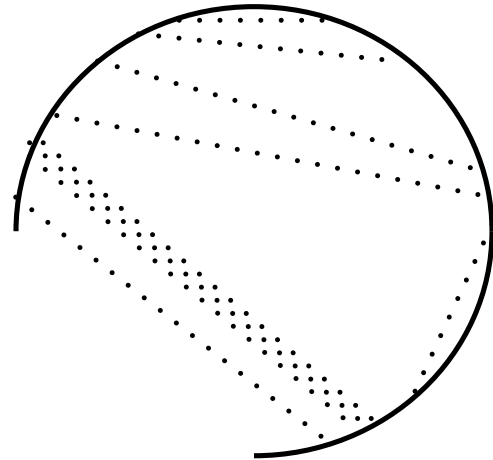
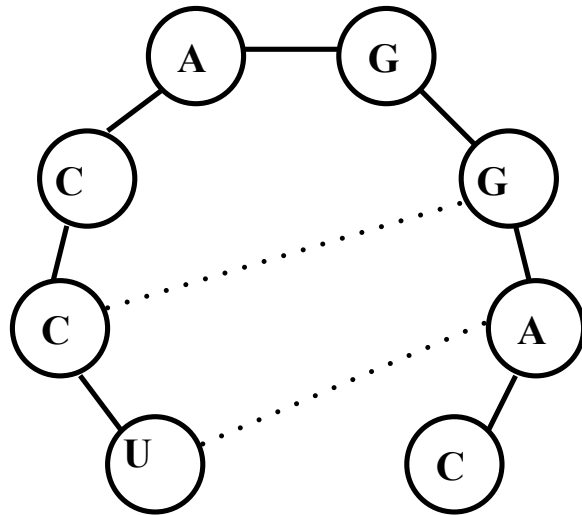
- Bases pair in order to form backbones and determine the secondary structure
- Aligning bases based on their ability to pair with each other gives an algorithmic approach to determining the optimal structure



Simplifying Assumptions

- RNA folds into one minimum free-energy structure.
 - There are no knots (base pairs never cross).
 - The energy of a particular base pair in a double stranded regions is sequence independent
 - Neighbors do not influence the energy.
 - Was solved by dynamic programming, Zuker and Stiegler 1981
-

Base Pair Maximization

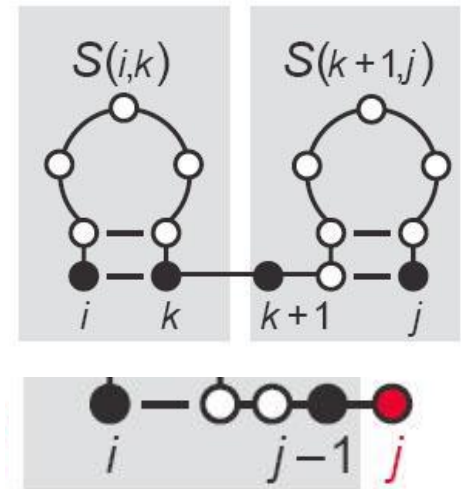


Base Pair Maximization – Dynamic Programming Algorithm

$S(i,j)$ is the folding of the subsequence of the RNA strand from index i to index j which results in the highest number of base pairs

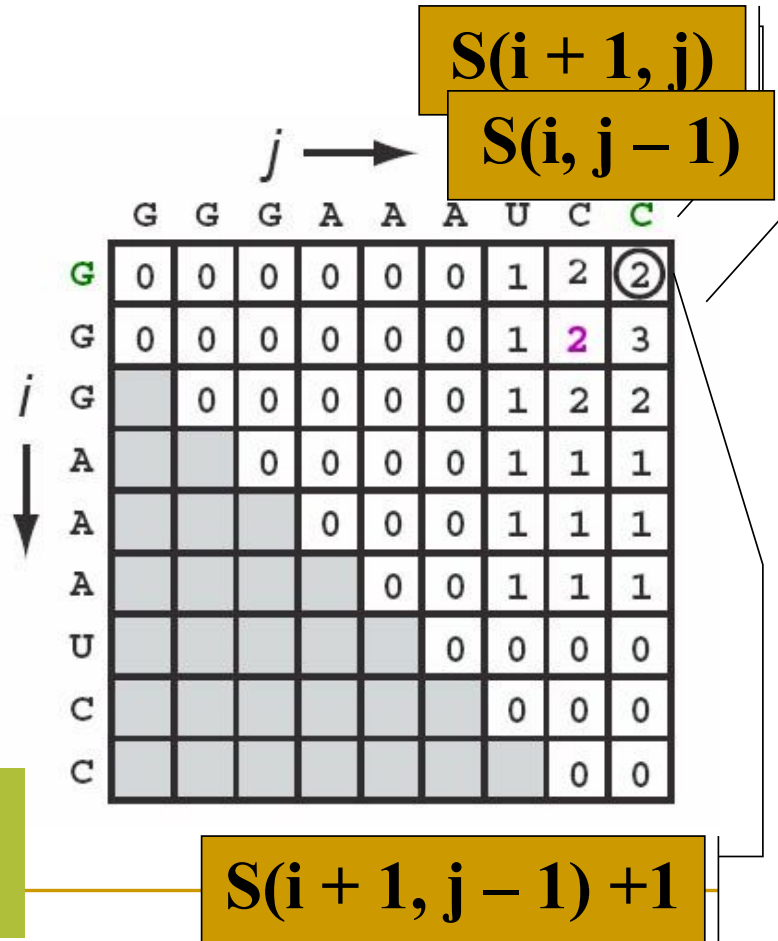
Maximizing Base

$$S(i,j) = \max \begin{cases} S(i+1, j-1) + 1 & \text{[if } i,j \text{ base pair]} \\ S(i+1, j) \\ S(i, j-1) \\ \max_{i < k < j} S(i, k) + S(k+1, j) \end{cases}$$



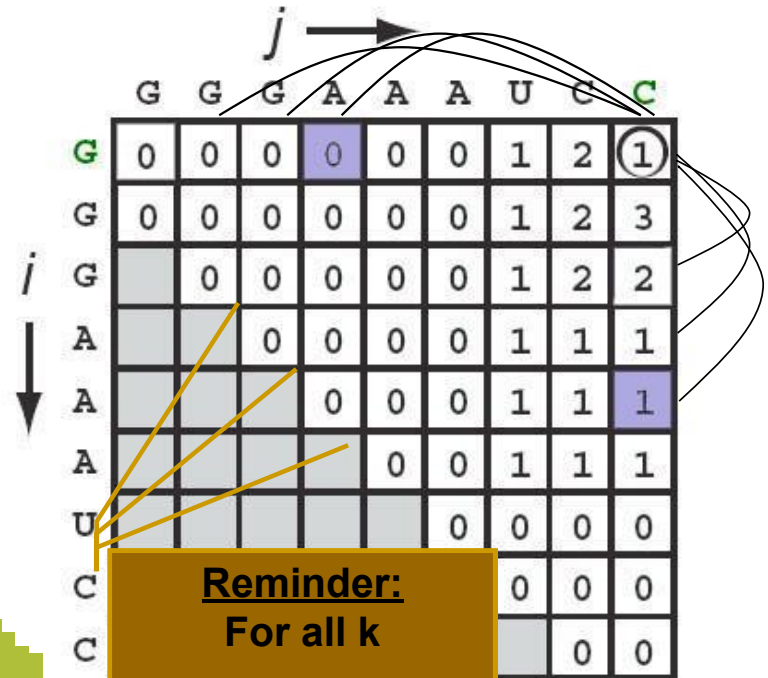
Base Pair Maximization – Dynamic Programming Algorithm

- **Alignment Method**
 - Align RNA strand to itself
 - Score increases for feasible base pairs
- Each score independent of overall structure
- Bifurcation adds extra dimension



Base Pair Maximization – Dynamic Programming Algorithm

- **Alignment Method**
 - Align RNA strand to itself
 - Score increases for feasible base pairs
- Each score independent of overall structure
- Bifurcation adds extra dimension



**Bifurcation – add values
for all k**

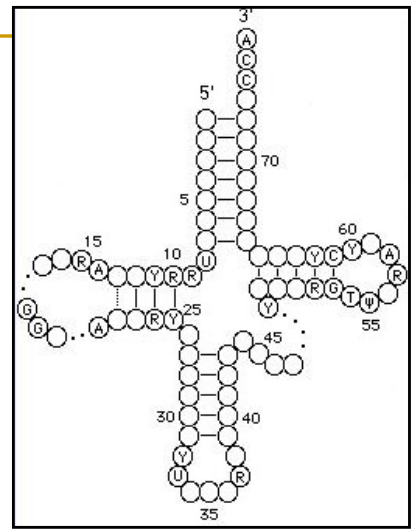
Reminder:
For all k
 $S(i,k) + S(k + 1, j)$

Base Pair Maximization - Drawbacks

- Base pair maximization will not necessarily lead to the most stable structure
 - May create structure with many interior loops or hairpins which are energetically unfavorable
 - Comparable to aligning sequences with scattered matches – not biologically reasonable
-

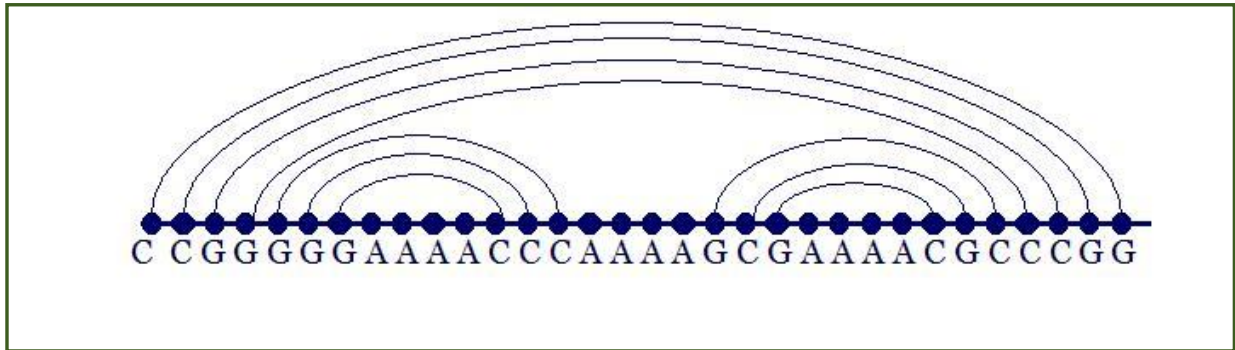
Energy Minimization

- Thermodynamic Stability
 - Estimated using experimental techniques
 - Theory : Most Stable is the Most likely
- No Pseudoknots due to algorithm limitations
- Uses Dynamic Programming alignment technique
- Attempts to maximize the score taking into account thermodynamics
- MFOLD and ViennaRNA



Free energy model

Free energy of a structure is the sum of all interactions energies

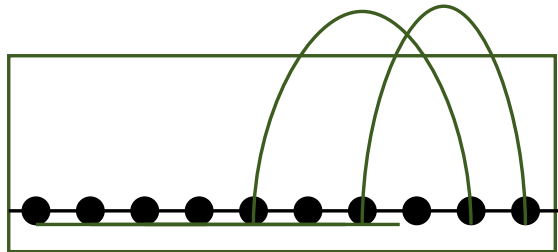


Free Energy(E) = E(CG)+E(CG)+.....

Each interaction energy can be calculated thermodynamically

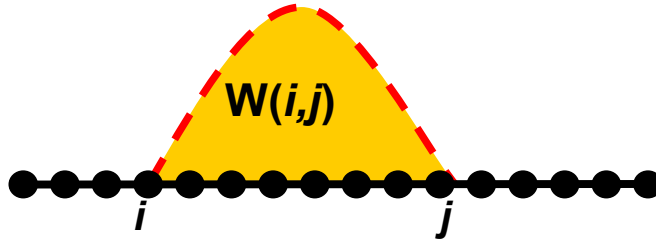
Why is MFE secondary structure prediction hard?

- MFE structure can be found by calculating free energy of all possible structures
- BUT the number of potential structures grows exponentially with the number, n , of bases



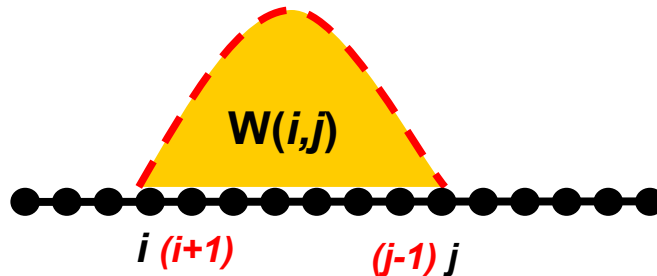
RNA folding with Dynamic programming (Zuker and Stiegler)

- **$W(i,j)$** : MFE structure of substrand from i to j



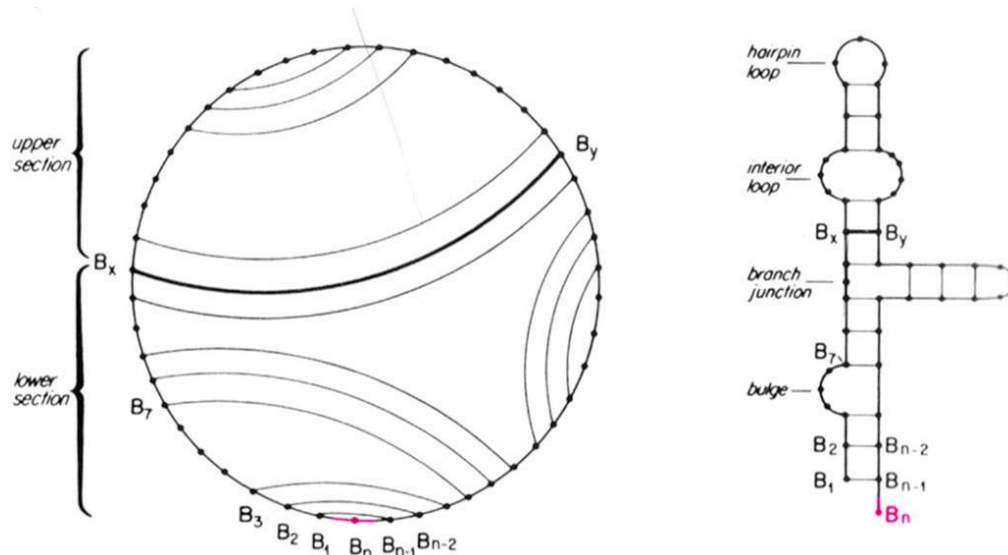
RNA folding with dynamic programming

- Assume a function $W(i,j)$ which is the MFE for the sequence starting at i and ending at j ($i < j$)



- Define scores, for example base pair (CG) = -1 non-pair(CA) = 1 (we want a negative score)
- Consider 4 possibilities:
 - i, j are a base pair, added to the structure for $i+1..j-1$
 - i is unpaired, added to the structure for $i+1..j$
 - j is unpaired, added to the structure for $i..j-1$
 - i, j are paired, but not to each other;
- Choose the minimal energy

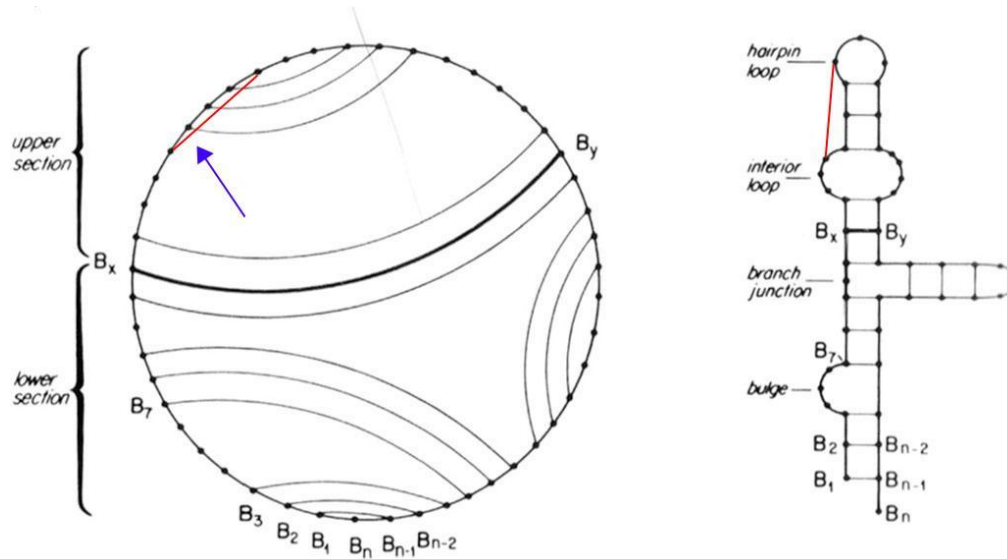
Energy Minimization Results



- All loops must have at least 3 bases in them
Equivalent to having 3 base pairs between all arcs

Exception: Location where the beginning and end of RNA come together in circularized representation

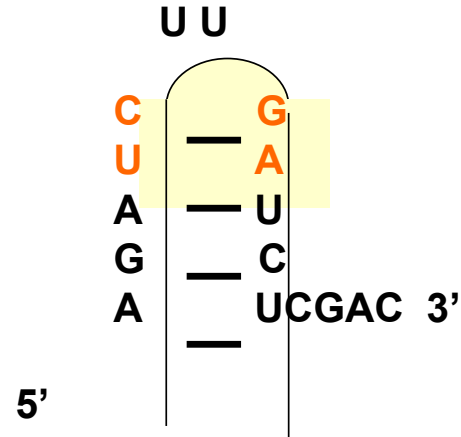
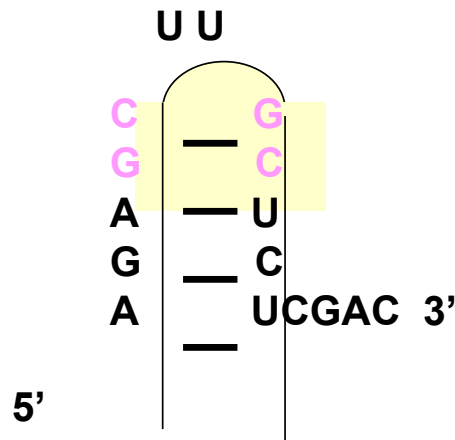
Trouble with Pseudoknots



- Pseudoknots cause a breakdown in the Dynamic Programming Algorithm.
- In order to form a pseudoknot, checks must be made to ensure base is not already paired – this breaks down the recurrence relations

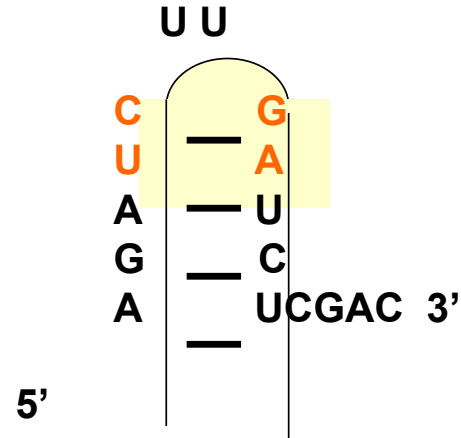
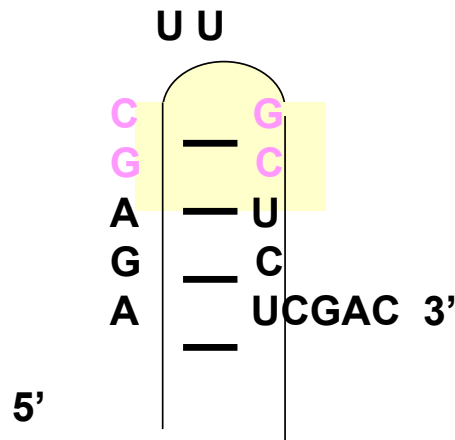
Sequence dependent free-energy

Nearest Neighbor Model



Energy is influenced by the previous base pair
(not by the base pairs further down).

Sequence dependent free-energy values of the base pairs



These energies are estimated experimentally from small synthetic RNAs.

Example values:

GC	GC	GC	GC
AU	GC	CG	UA
-2.3	-2.9	-3.4	-2.1

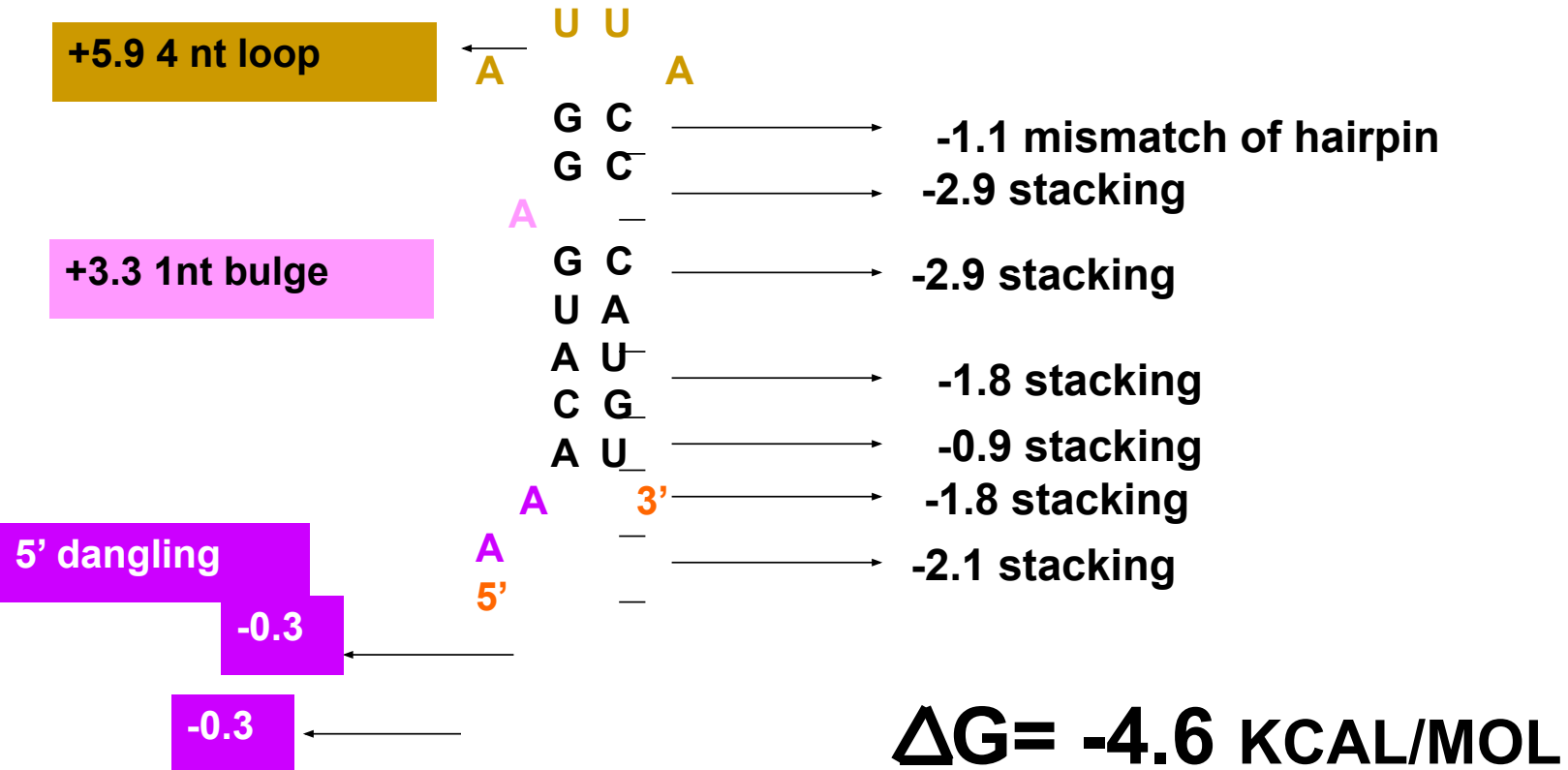
Adding Complexity to Energy Calculations

- Stacking energy - Assign negative energies to these *between base pair* regions.
 - Energy is influenced by the previous base pair (not by the base pairs further down).
 - These energies are estimated experimentally from small synthetic RNAs.
- Positive energy - added for destabilizing regions such as bulges, loops, etc.
- More than one structure can be predicted

Mfold

- Positive energy - added for destabilizing regions such as bulges, loops, etc.
- More than one structure can be predicted

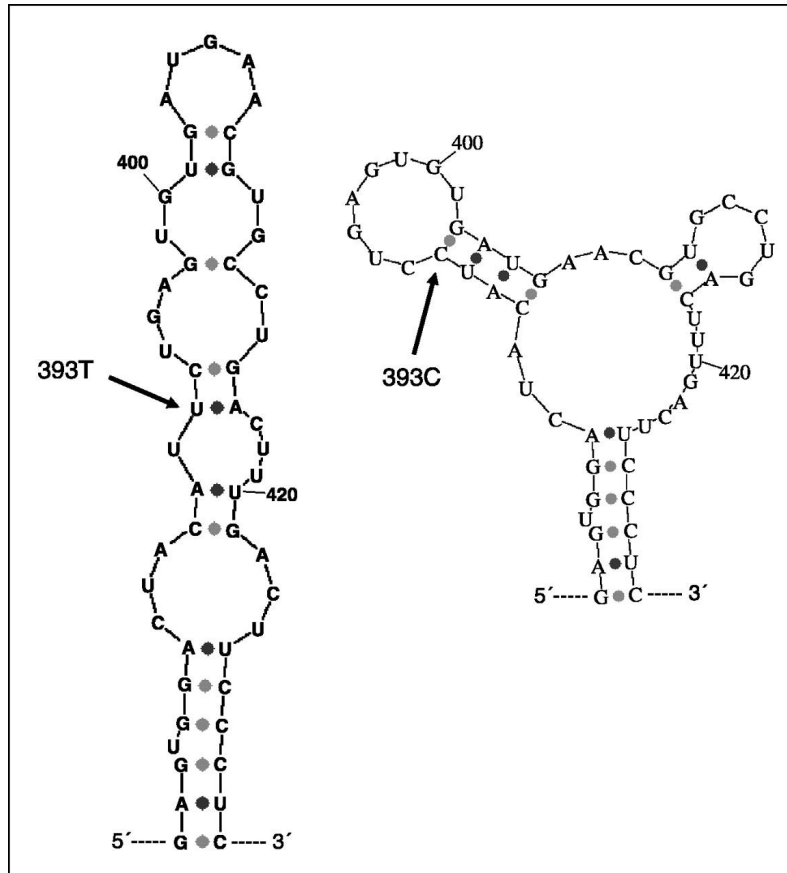
Free energy computation



Mfold

- Positive energy - added for destabilizing regions such as bulges, loops, etc.
 - More than one structure can be predicted
-

More than one structure can be predicted for the same RNA



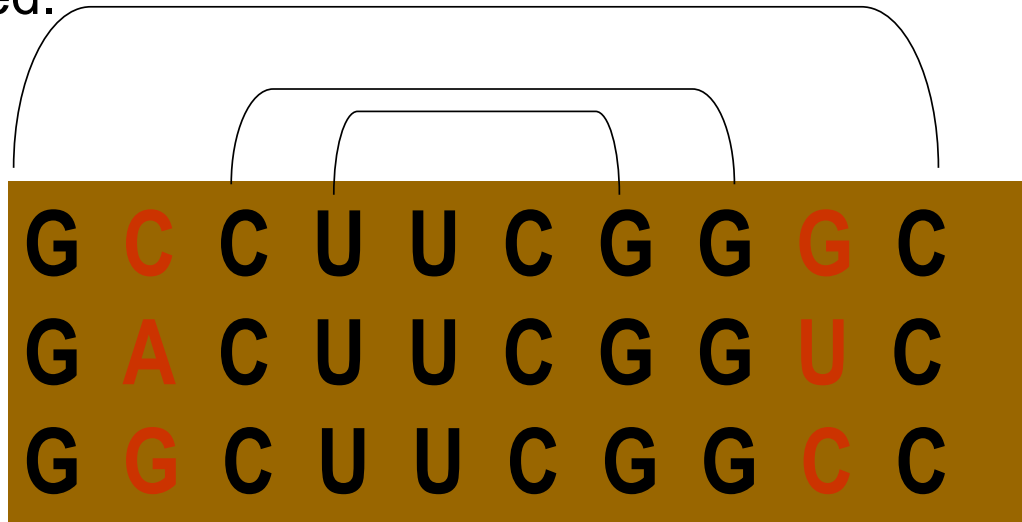
Frey U H et al. Clin Cancer Res 2005;11:5071-5077

Energy Minimization Drawbacks

- Compute only one optimal structure
 - Usual drawbacks of purely mathematical approaches
 - Similar difficulties in other algorithms
 - Protein structure
 - Exon finding
-

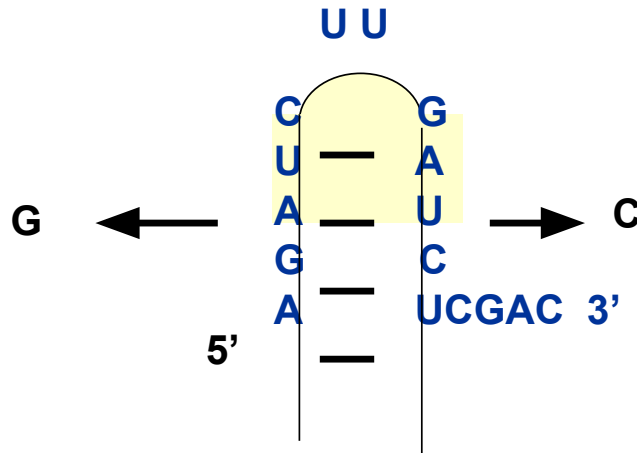
RNA fold prediction based on Multiple Alignment

Information from multiple sequence alignment (MSA) can help to predict the probability of positions i, j to be base-paired.

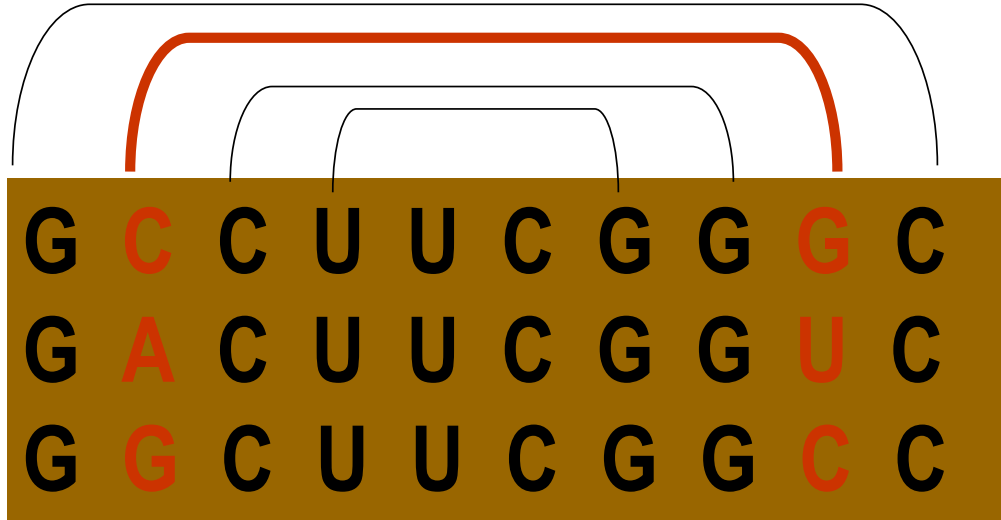


Compensatory Substitutions

Mutations that maintain the secondary structure can help predict the fold



RNA secondary structure can be revealed by
identification of compensatory mutations



U C
U C
N G
G C

Insight from Multiple Alignment

Information from multiple sequence alignment (MSA) can help to predict the probability of positions i, j to be base-paired.

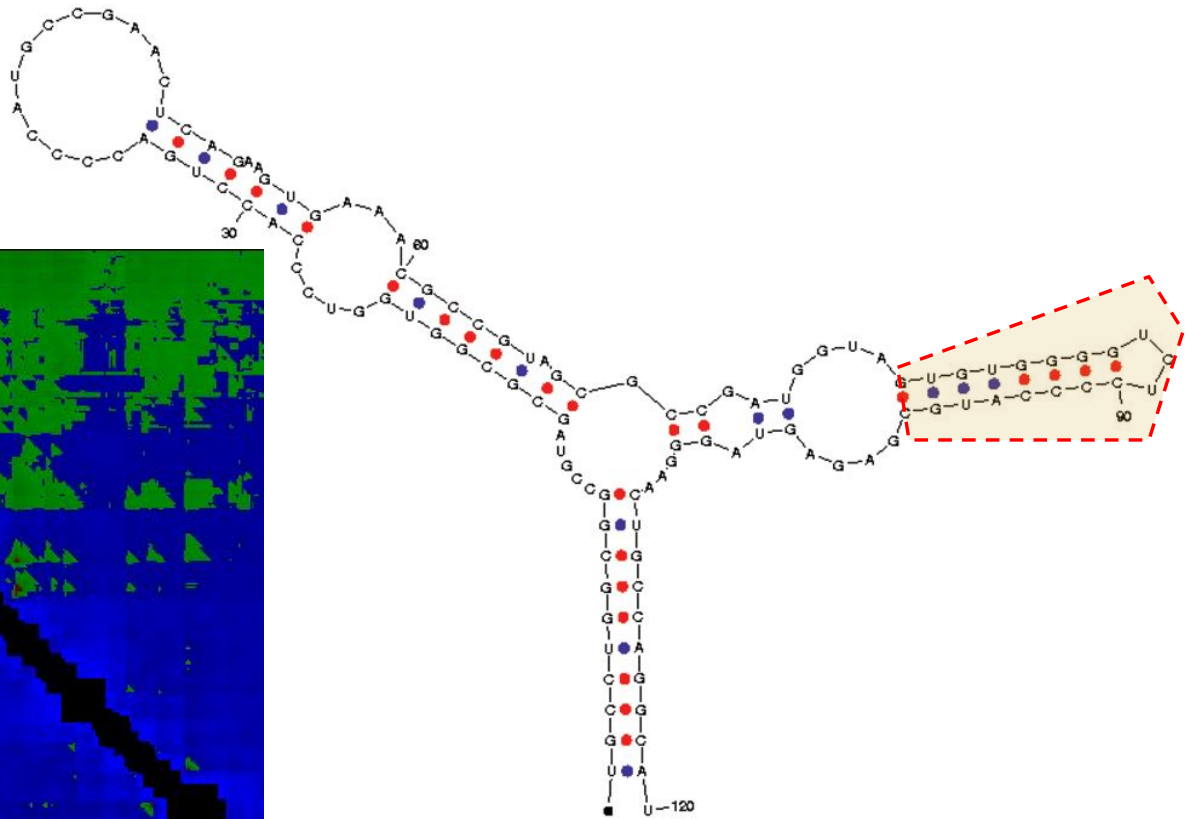
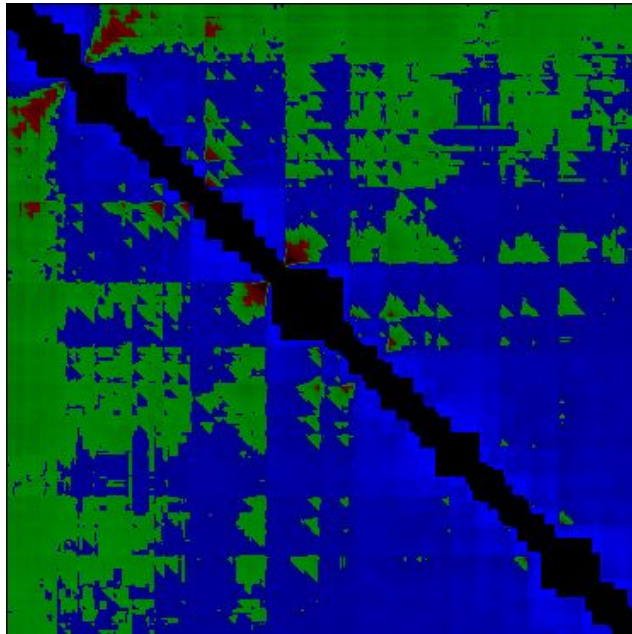
- Conservation – no additional information
 - Consistent mutations (GC \square GU) – support stem
 - Inconsistent mutations – does not support stem.
 - Compensatory mutations – support stem.
-

RNAalifold

- Predicts the consensus secondary structure for a set of aligned RNA sequences by using modified dynamic programming algorithm that add alignment information to the standard energy model
 - Improvement in prediction accuracy
-

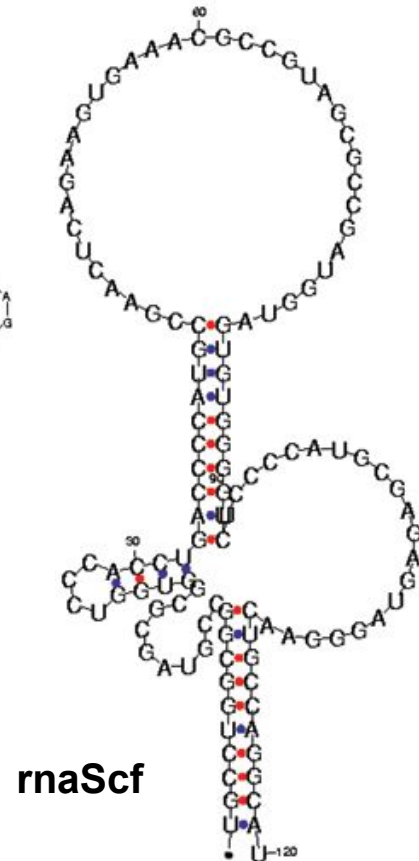
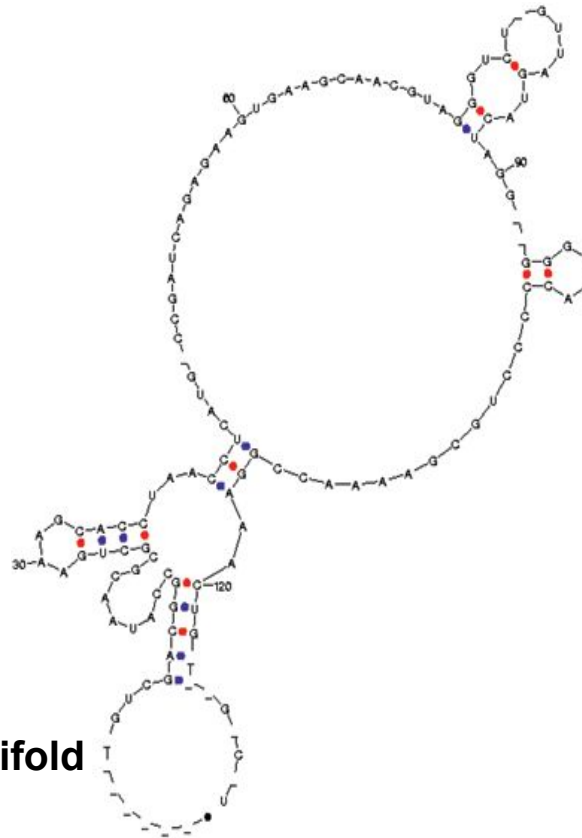
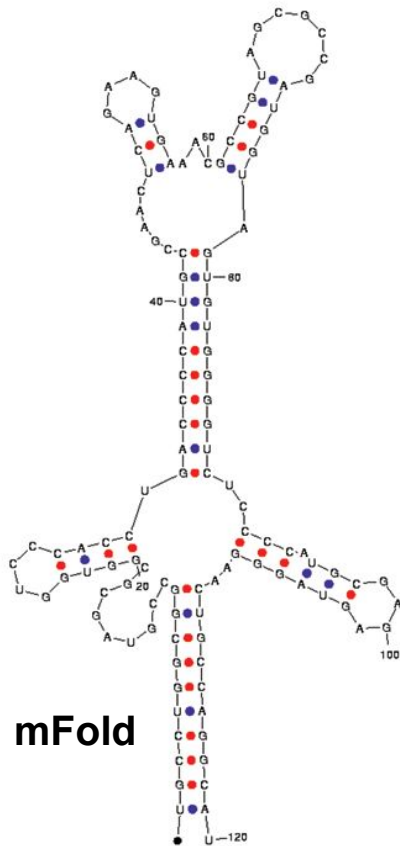
LOCAL MINIMUM FREE ENERGY: DENSITYFOLD

E.coli 5S rRNA



Energy Density Landscape

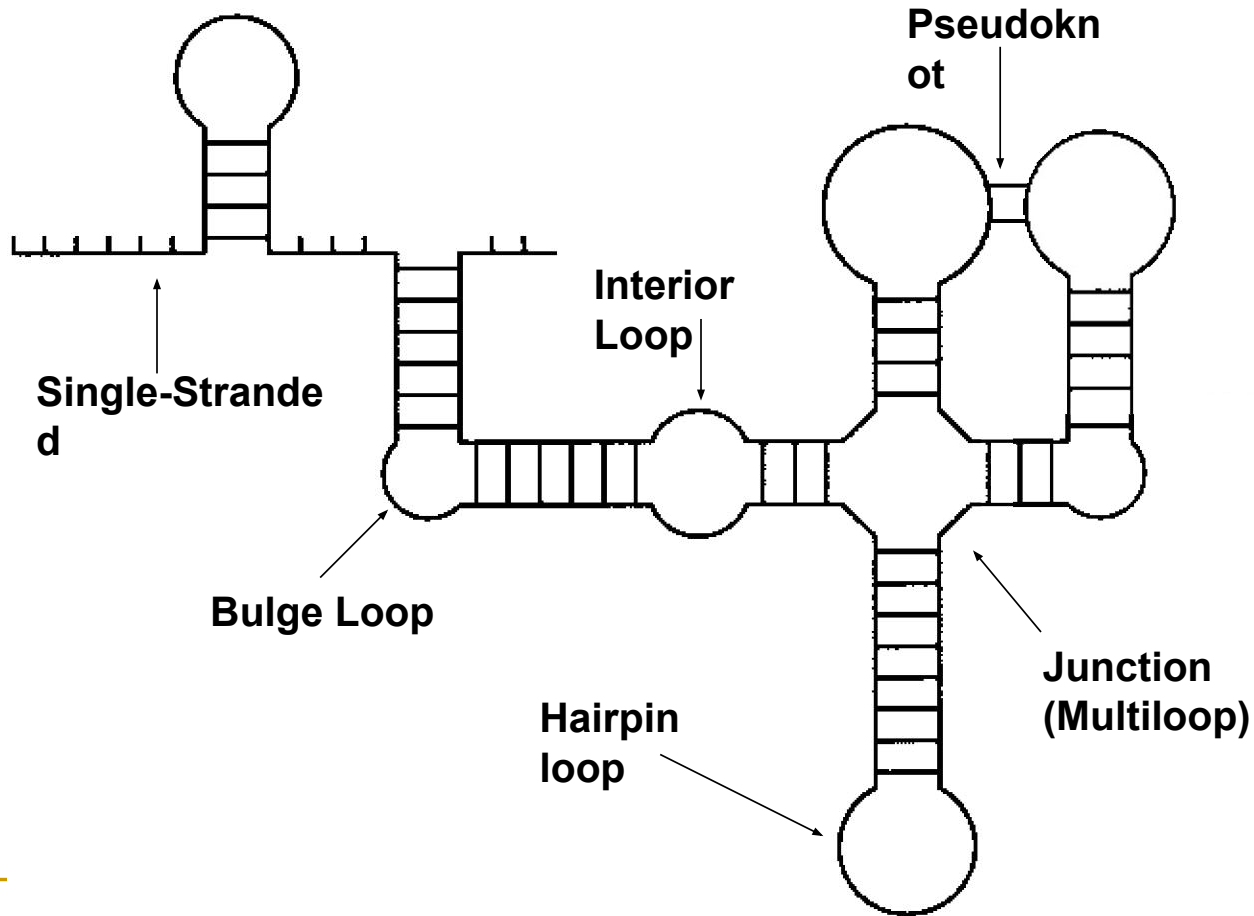
E.coli 5S rRNA predictions



Densityfold (alteRNA)

- Instead of finding minimum global free energy, find local minimum free energies
 - Emulate the folding process of RNA folding by aiming to keep locally stable substructures
 - *Energy density seen by a basepair*: the free energy of the “optimal substructure” normalized by distance
 - *Energy density of an unpaired base*: energy density of the nearest encapsulating basepair
 - Densityfold optimizes a linear combination of free energy and total energy density
 - For every potential basepair, compute the optimal contribution of the implied substructure
 - The optimization function is non linear
 - Hill climbing process for approximating the contributions of unpaired bases
-

Substructures



Densityfold energy types

- $eH(i,j,)$: free energy of a hairpin loop enclosed by the base pair $S[i].S[j]$
- $eS(i,j,)$: free energy of the base pair $S[i].S[j]$ provided that it forms a stacking pair with $S[i+1].S[j-1]$
- $eBI(i,j,i',j')$: free energy of an internal loop or a bulge that starts with $S[i].S[j]$ and ends with $S[i'].S[j']$

Densityfold energy types

- $eM(i,j,i_1,j_1,\dots,i_k,j_k)$: free energy of multibranch loop that starts with $S[i].S[j]$ and branches out $S[i_1].S[j_1]$, $S[i_2].S[j_2]$, ..., $S[i_k].S[j_k]$
 - $eDA(j,j-1)$: free energy of an unpaired dangling base $S[j]$ when $S[j-1]$ forms a base pair with another base
-

Densityfold energy tables

- $ED(j)$: minimum total free energy density of a secondary structure for substring $S[1, j]$.
- $E(j)$: free energy of the energy density minimized secondary structure for substring $S[1, j]$.
- $ED_s(i, j)$: minimum total free energy density of a secondary structure for $S[i, j]$, provided that $S[i].S[j]$ is a base pair.
- $E_s(i, j)$: free energy of the energy density minimized secondary structure for the substring $S[i, j]$, provided that $S[i].S[j]$ is a base pair.

Densityfold energy tables

- $ED_{B_I}(i, j)$: minimum total free energy density of a secondary structure for $S[i, j]$, provided that there is a bulge or an internal loop starting with base pair $S[i].S[j]$.
- $E_{B_I}(i, j)$: free energy of an energy density minimized structure for $S[i, j]$, provided that a bulge or an internal loop starting with base pair $S[i].S[j]$.
- $ED_M(i, j)$: minimum total free energy density of a secondary structure for $S[i, j]$, such that there is a multibranch loop starting with base pair $S[i].S[j]$.
- $E_M(i, j)$: free energy of an energy density minimized structure for $S[i, j]$, provided there is a multibranch loop starting with base pair $S[i].S[j]$.

Calculating energy tables

$$ED(j) = \min \left\{ ED(j-1), \min_{1 \leq i \leq j-1} \{ED(i-1) + ED_S(i, j)\} \right\}$$

$$ED_S(i, j) = \min \left\{ \begin{array}{l} +\infty, \\ eH(i, j), \\ 2 \frac{eS(i, j) + E_S(i+1, j-1)}{j-i+1} + ED_S(i+1, j-1), \\ ED_{BI}(i, j), \\ ED_M(i, j) \end{array} \begin{array}{l} (i) \\ (ii) \\ (iii) \\ (iv) \\ (v) \end{array} \right\}$$

- Similar calculations for other tables
- $O(n^{k+2})$ time and $O(n^2)$ space

Linear combination of MFE and ED

For any $x \in \{S, BI, M\}$ let $ELC_x(i, j) = ED_x(i, j) + E_x(i, j)$.

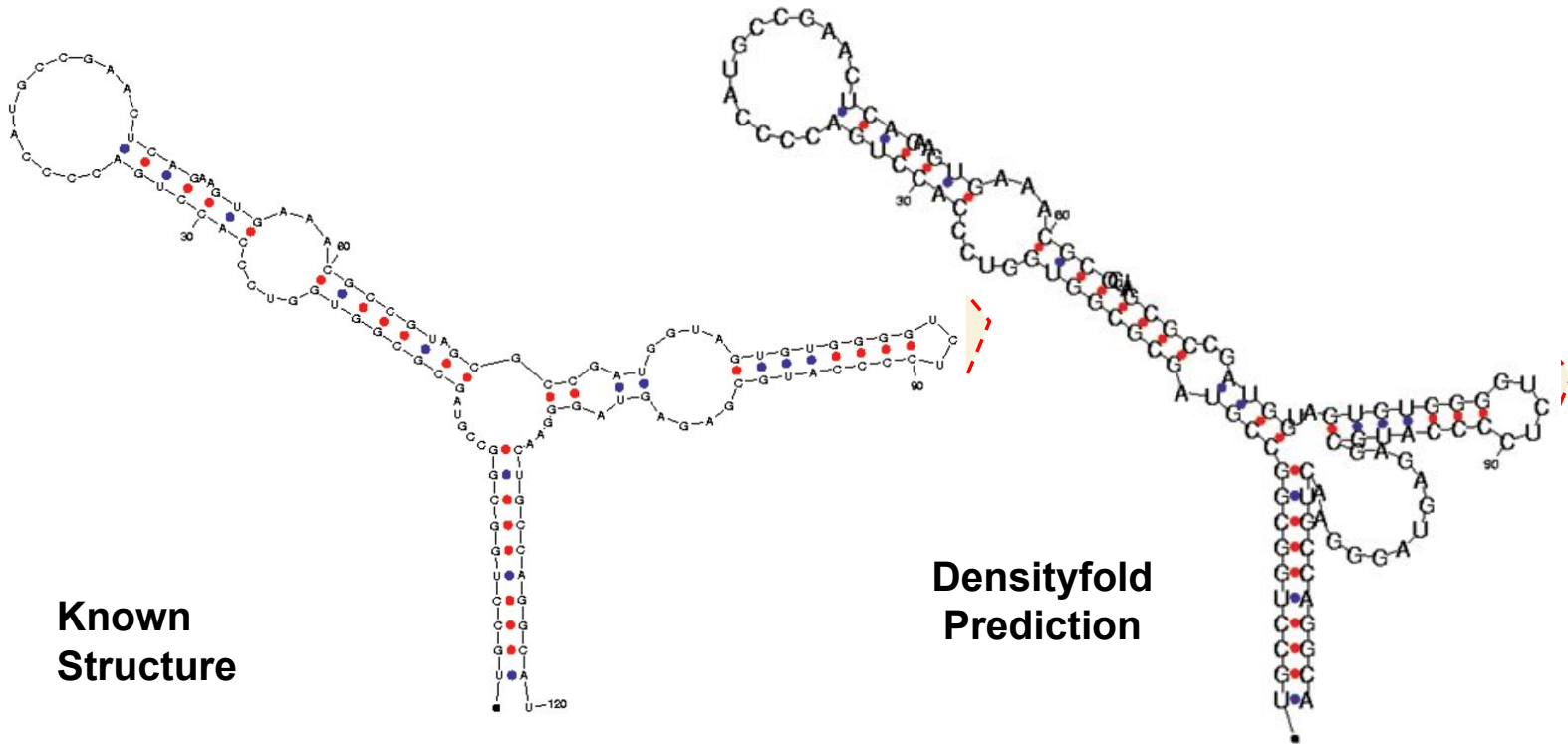
Optimize $ELC(n) = ED(n) + E(n)$.

$$ELC(j) = \min \left\{ ELC(j-1), \min_{1 \leq i \leq j-1} \{ELC(i-1) + ELC_S(i, j)\} \right\}$$

$$ELC_S(i, j) = \min \left\{ \begin{array}{ll} +\infty, & (i) \\ eH(i, j) \cdot (1 + \sigma), & (ii) \\ 2 \frac{eS(i, j) + E_S(i+1, j-1)}{j-i+1} + ELC_S(i+1, j-1) + \sigma \cdot eS(i, j), & (iii) \\ ELC_{BI}(i, j), & (iv) \\ ELC_M(i, j) & (v) \end{array} \right\}$$

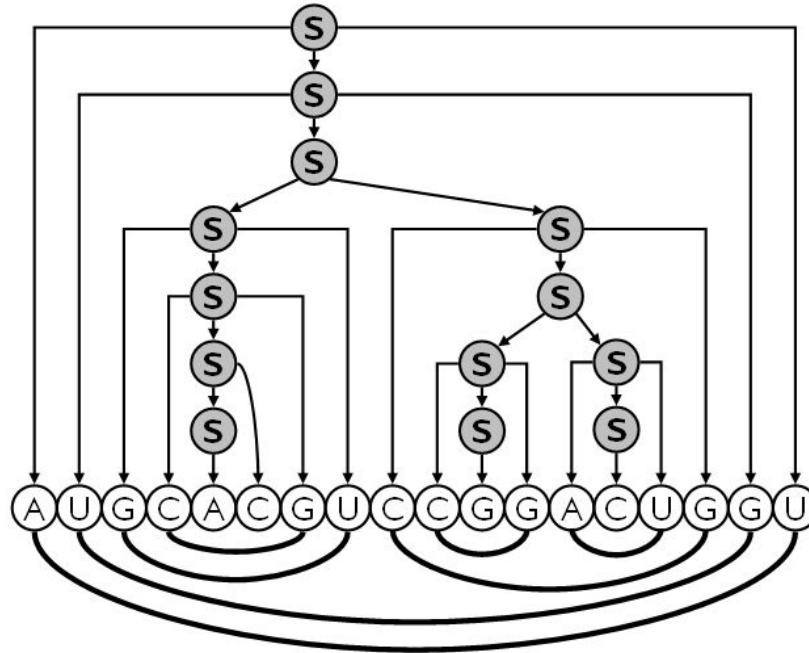
- Similar formulations for ELC_{BI} and ELC_M
- $O(n^4)$ running time

Densityfold prediction: E.coli 5S rRNA



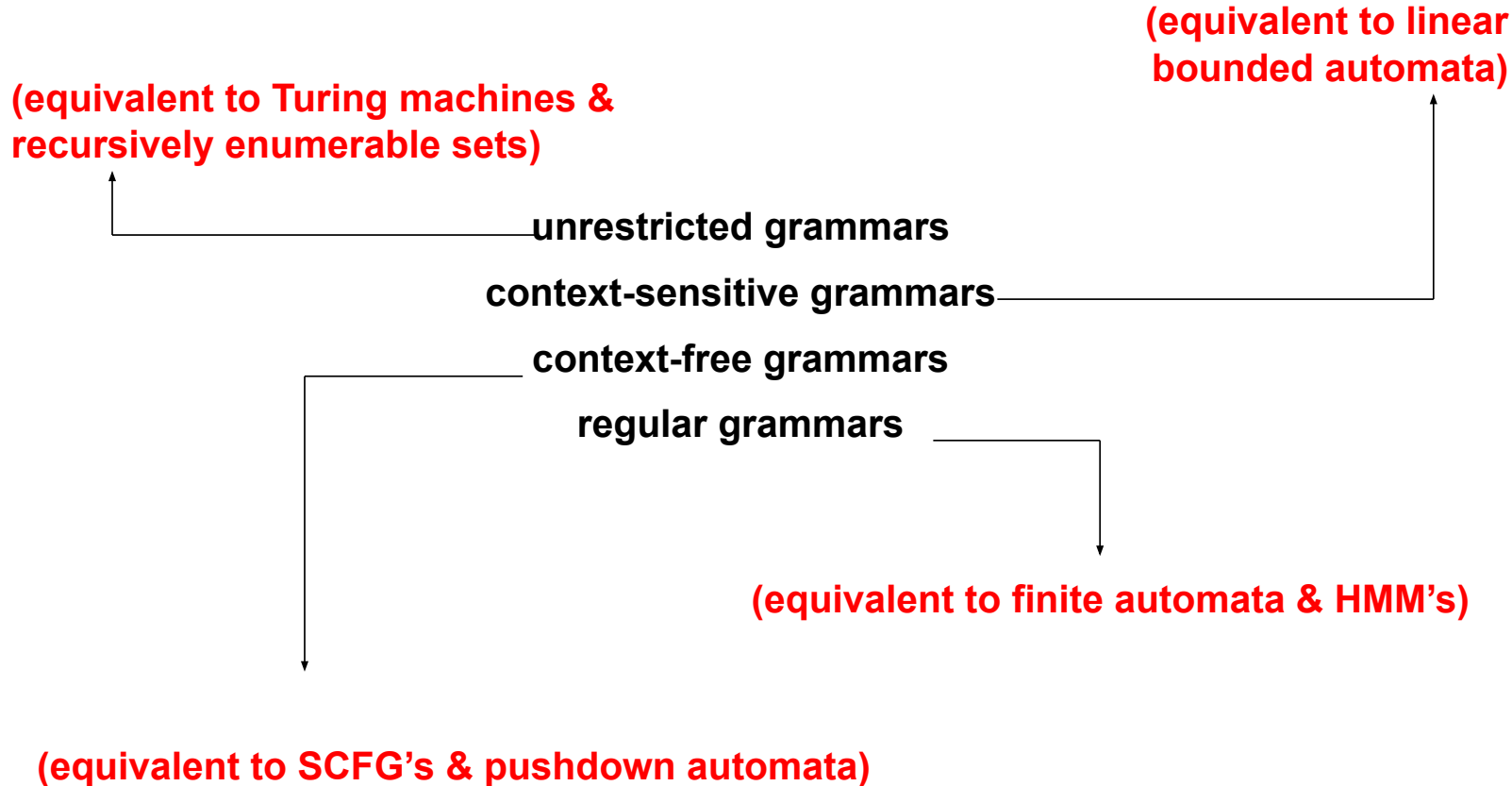
STOCHASTIC CONTEXT-FREE GRAMMARS

SCFG



- RNA folding can be represented as context-free grammars

Chomsky hierarchy



Context-free grammars

A *context-free grammar* is a generative model denoted by a 4-tuple:

$$G = (V, \Sigma, S, P)$$

where:

Σ is a *terminal alphabet*, (e.g., $\{a, c, g, u\}$)

V is a *nonterminal alphabet*, (e.g., $\{A, B, C, D, E, \dots\}$)

$S \in V$ is a special *start symbol*, and

P is a set of rewriting rules called *productions*.

Productions in P are rules of the form:

$$X \rightarrow \lambda$$

where $X \in V, \lambda \in (V \cup \alpha)^*$

Context “freeness”

The “*context-freeness*” is imposed by the requirement that the l.h.s of each production rule may contain only a single symbol, and that symbol must be a nonterminal:

$$X \rightarrow \lambda$$

Thus, a CFG cannot specify *context-sensitive* rules such as:

$$wXz \rightarrow w\lambda z$$

Derivations

Suppose a CFG G has generated a *terminal string* $x \in \Sigma^*$. A *derivation* $S \Rightarrow^* x$ denotes a possible derivation for generating x .

A *derivation* (or *parse*) consists of a series of applications of productions from P , beginning with the *start symbol* S and ending with the *terminal string* x :

$$S \Rightarrow s_1 \Rightarrow s_2 \Rightarrow s_3 \Rightarrow \dots \Rightarrow x$$

where $s_i \in (V \cup \Sigma)^*$.

We'll concentrate on leftmost derivations where the leftmost nonterminal is always replaced first.

Context-free vs. regular

The advantage of CFG's over HMM's lies in their ability to model arbitrary runs of matching pairs of elements, such as matching pairs of parentheses:

$$L(((((((L)))))))))L$$

When the number of matching pairs is unbounded, a finite-state model such as a DFA or an HMM is inadequate to enforce the constraint that all left elements must have a matching right element.

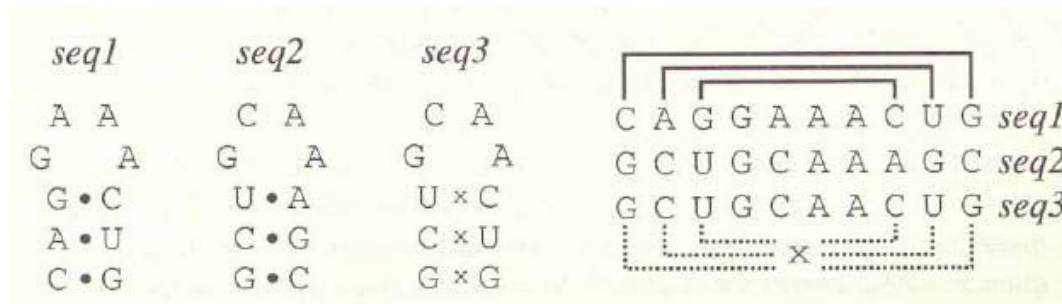
In contrast, in a CFG we can use rules such as $X \rightarrow (X)$. A sample derivation using such a rule is:

$$X \Rightarrow (X) \Rightarrow ((X)) \Rightarrow (((X))) \Rightarrow ((((X)))) \Rightarrow ((((((X))))))$$

An additional rule such as $X \rightarrow \epsilon$ is necessary to terminate the recursion.

A CFG for an RNA

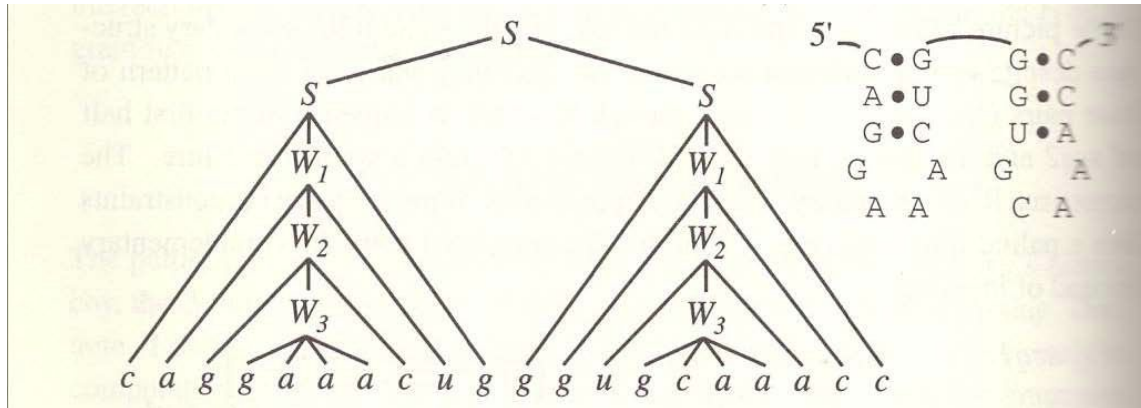
- RNA hairpin with 3 bp stem and a 4-base loop (GAAA or GCAA)



S-> aXu | cXg | gXc | uXa
X-> aYu | cYg | gYc | uYa
Y-> aZu | cZg | gZc | uZa
Z->gaaa | gcaa

Parse trees

- A representation of a parse of a string by a CFG
- **Root** – start nonterminal S
- **Leaves** – terminal symbols in the given string
- **Internal nodes** - nonterminals
- The children of an internal node are the productions of that nonterminal (left-to-right order)



Stochastic CFG

A *stochastic context-free grammar* (*SCFG*) is a CFG plus a probability distribution on productions:

$$G = (V, \Sigma, S, P, R_p)$$

where $P_p : P \rightarrow [0, 1]$, and probabilities are normalized at the level of each l.h.s. symbol X :

$$\forall X \in V \left[\sum_{X \rightarrow \lambda} R_p(X \rightarrow \lambda) = 1 \right]$$

Thus, we can compute the probability of a single derivation $S \Rightarrow^* x$ by multiplying the probabilities for all productions used in the derivation:

$$\prod_i R(X_i \rightarrow \lambda_i)$$

We can sum over all possible (leftmost) derivations of a given string x to get the probability that G will generate x at random:

$$R(x | G) = \sum_j R(S \Rightarrow_j^* x | G).$$

An example

As an example, consider $G=(V_G, \alpha, S, P_G, R_G)$, for $V_G=\{S, L, N\}$, $\Sigma=\{a, c, g, t\}$, and P_G the set consisting of:

$$S \rightarrow a S u \mid u S a \mid c S g \mid g S c \mid L \quad (P=0.2)$$

$$L \rightarrow N N N N \quad (P=1.0)$$

$$N \rightarrow a \mid c \mid g \mid u \quad (P=0.25)$$

Then the probability of the sequence $acguacguacgu$ is given by:

$$\begin{aligned} P(acguacguacgu) = \\ P(S \Rightarrow aSu \Rightarrow acSgu \Rightarrow acgScgu \Rightarrow acguSacgu \Rightarrow \\ acguLacgu \Rightarrow acguNNNNacgu \Rightarrow acguaNNNacgu \Rightarrow \\ acguacNNacgu \Rightarrow acguacgNacgu \Rightarrow acguacguacgu) = \\ 0.2 \times 0.2 \times 0.2 \times 0.2 \times 0.2 \times 1 \times 0.25 \times 0.25 \times 0.25 \times 0.25 = 1.25 \times 10^{-6} \end{aligned}$$

because this sequence has only one possible (leftmost) derivation under grammar G .

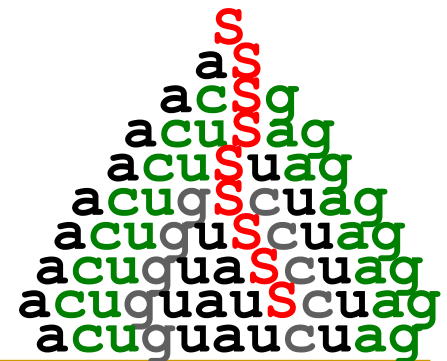
Structure using SFCG

- Grammar rules with associated probabilities

S	□	aSu		cSg		aS		uS		...		Su		SS		ε
<i>P</i>		.21		.15		.11		.08				.03		.22		.02

- We select the set of transformations that highest probability of generating the input sequence. This set gives us our structure.
- Let's generate a structure for the sequence acuguaucuaag

acuguaucuaag
- ((X((X)))))



Chomsky Normal Form

A CNF grammar is one in which all productions are of the form:

$$X \rightarrow YZ$$

or:

$$X \rightarrow a$$

Non-CNF:

$$S \rightarrow aSt | tSa | cSg | gSc | L$$

$$L \rightarrow NNNN$$

$$N \rightarrow a | c | g | u$$

CNF:

$$S \rightarrow AS_T | TS_A | CS_G | GS_C | NL_1$$

$$S_A \rightarrow SA$$

$$S_T \rightarrow ST$$

$$S_C \rightarrow SC$$

$$S_G \rightarrow SG$$

$$L_1 \rightarrow NL_2$$

$$L_2 \rightarrow NN$$

$$N \rightarrow a | c | g | u$$

$$A \rightarrow a$$

$$C \rightarrow c$$

$$G \rightarrow g$$

$$T \rightarrow u$$

Parsing CFG

Two questions for a CFG:

- 1) Can a grammar G derive string x ?
- 2) If so, what series of productions would be used during the derivation? (*there may be multiple answers!*)

Additional questions for an SCFG:

- 1) What is the *probability* that G derives string x ?
- 2) What is the *most probable* derivation of x via G ?

Parsing CFG

- CYK Algorithm (Cocke-Younger-Kasami)
 - Dynamic Programming method
 - Modified CYK for SCFG
 - “Inside algorithm”
 - Training similar to HMM
 - If parses are known for training data sequences, simply count the number of times for each production, calculate probabilities (labeled sequence training for HMM)
 - If parses are not known, apply an EM algorithm called “Inside-Outside” (“forward-backward” for HMM)
-