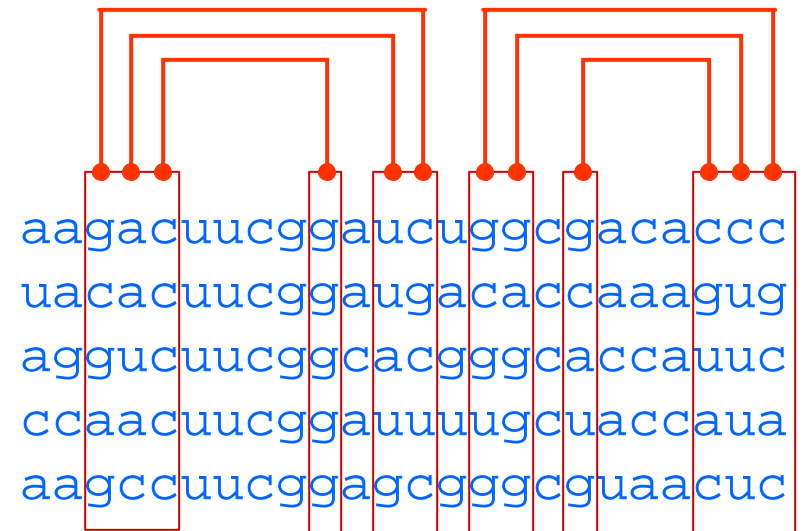


RNA Secondary Structure



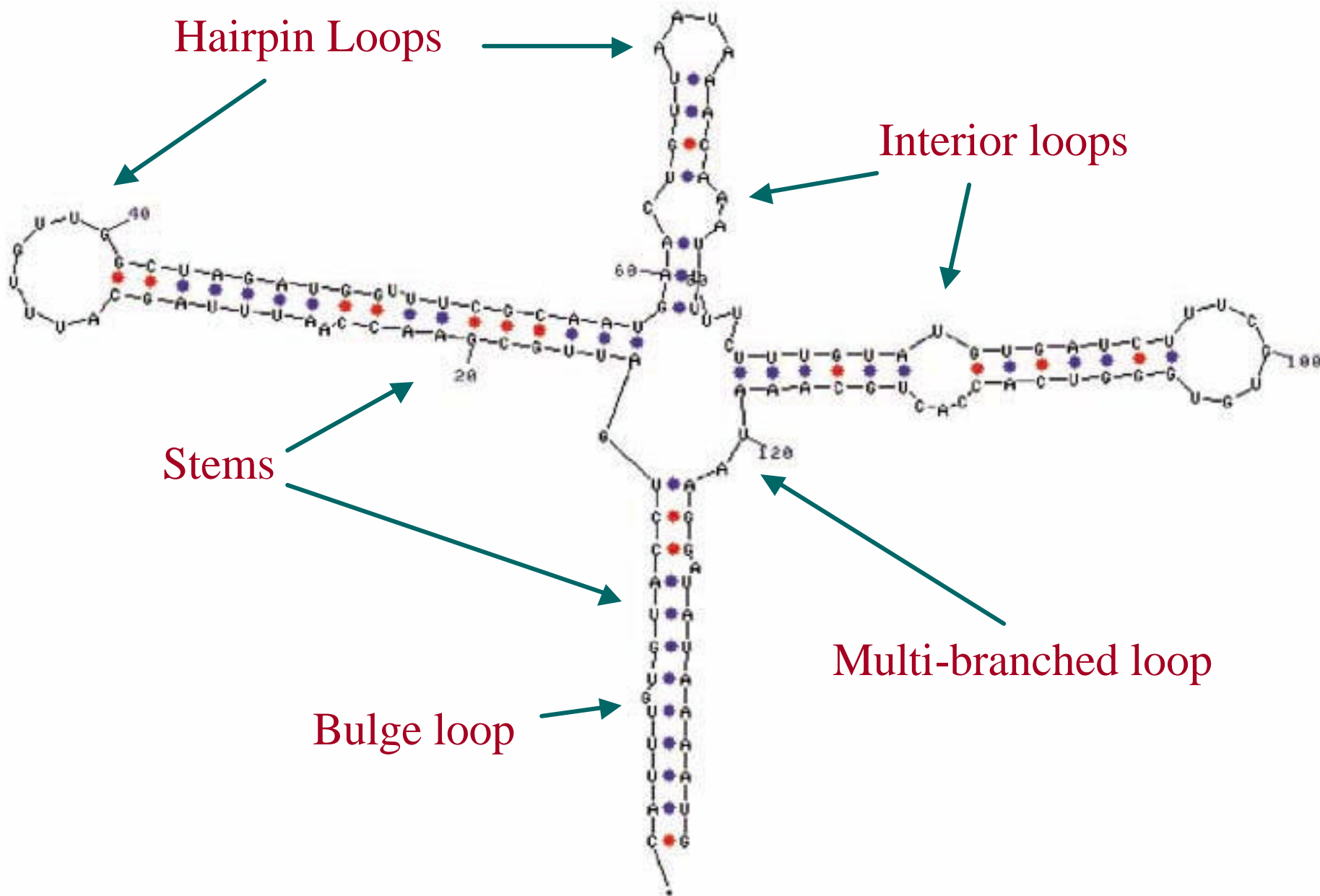
Hairpin Loops

Interior loops

Stems

Multi-branched loop

Bulge loop



Context Free Grammars and RNAs

$S \rightarrow a W_1 u$

$W_1 \rightarrow c W_2 g$

$W_2 \rightarrow g W_3 c$

$W_3 \rightarrow g L c$

$L \rightarrow agucg$

ACGG AG
UGCC U
CG

What if the stem loop can have other letters in place of the ones shown?

The Nussinov Algorithm and CFGs

Define the following grammar, with scores:

$$\begin{array}{lcl} S \rightarrow a S u : 3 & | & u S a : 3 \\ & & g S c : 2 \quad | \quad c S g : 2 \\ & & g S u : 1 \quad | \quad u S g : 1 \end{array}$$
$$S S : 0 \quad |$$
$$a S : 0 \quad | \quad c S : 0 \quad | \quad g S : 0 \quad | \quad u S : 0 \quad | \quad \varepsilon : 0$$

Note: ε is the "" string

Then, the Nussinov algorithm finds the optimal parse of a string with this grammar

The Nussinov Algorithm

Initialization:

$F(i, i-1) = 0;$ for $i = 2$ to N

$F(i, i) = 0;$ for $i = 1$ to N

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

Iteration:

For $l = 2$ to N :

For $i = 1$ to $N - l$

$j = i + l - 1$

$$F(i, j) = \max \begin{cases} F(i+1, j-1) + s(x_i, x_j) & S \rightarrow a S u | \dots \\ \max_{i \leq k < j} \{ F(i, k) + F(k+1, j) \} & S \rightarrow S S \end{cases}$$

Termination:

Best structure is given by $F(1, N)$

Stochastic Context Free Grammars

In an analogy to HMMs, we can assign probabilities to transitions:

Given grammar

$$X_1 \rightarrow s_{11} \mid \dots \mid s_{in}$$

...

$$X_m \rightarrow s_{m1} \mid \dots \mid s_{mn}$$

Can assign probability to each rule, s.t.

$$P(X_i \rightarrow s_{i1}) + \dots + P(X_i \rightarrow s_{in}) = 1$$

Computational Problems

- Calculate an optimal alignment of a sequence and a SCFG

(DECODING)

- Calculate $\text{Prob}[\text{sequence} \mid \text{grammar}]$

(EVALUATION)

- Given a set of sequences, estimate parameters of a SCFG

(LEARNING)

Evaluation

Recall HMMs:

Forward: $f_l(i) = P(x_1 \dots x_i, \pi_i = l)$

Backward: $b_k(i) = P(x_{i+1} \dots x_N \mid \pi_i = k)$

Then,

$$P(x) = \sum_k f_k(N) a_{k0} = \sum_l a_{0l} e_l(x_1) b_l(1)$$

Analogue in SCFGs:

Inside: $a(i, j, V) = P(x_i \dots x_j \text{ is generated by nonterminal } V)$

Outside: $b(i, j, V) = P(x, \text{ excluding } x_i \dots x_j \text{ is generated by } S \text{ and the excluded part is rooted at } V)$

Normal Forms for CFGs

Chomsky Normal Form:

$$X \rightarrow YZ$$
$$X \rightarrow a$$

All productions are either to 2 nonterminals, or to 1 terminal

Theorem (technical)

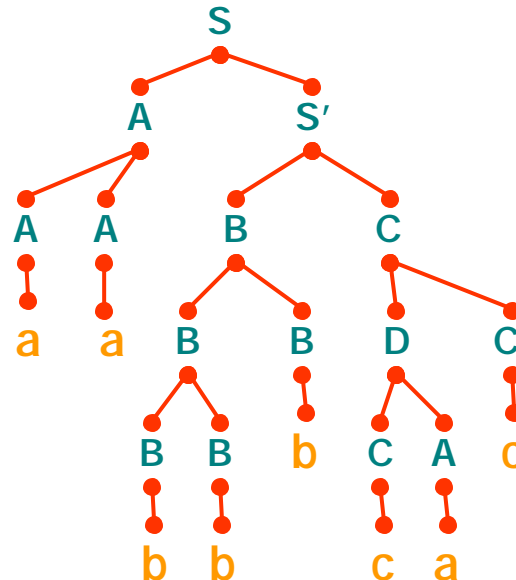
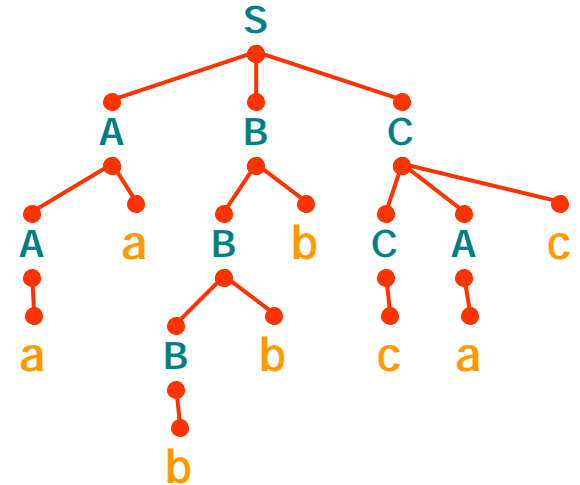
Every CFG has an equivalent one in Chomsky Normal Form

(That is, the grammar in normal form produces exactly the same set of strings)

Example of converting a CFG to C.N.F.

$$S \rightarrow ABC$$
$$A \rightarrow Aa \mid a$$
$$B \rightarrow Bb \quad | \quad b$$
$$C \rightarrow CAc \quad | \quad c$$

Converting:

$$S \rightarrow AS'$$
$$S' \rightarrow BC$$
$$A \rightarrow AA \mid a$$
$$B \rightarrow BB \mid b$$
$$C \rightarrow DC' \mid c$$
$$C' \rightarrow C$$
$$D \rightarrow CA$$


Another example

$S \rightarrow ABC$

$A \rightarrow C \mid aA$

$B \rightarrow bB \mid b$

$C \rightarrow cCd \mid c$

Converting:

$S \rightarrow AS'$

$S' \rightarrow BC$

$A \rightarrow C'C'' \mid c \mid A'A$

$A' \rightarrow a$

$B \rightarrow B'B \mid b$

$B' \rightarrow b$

$C \rightarrow C'C'' \mid c$

$C' \rightarrow c$

$C'' \rightarrow CD$

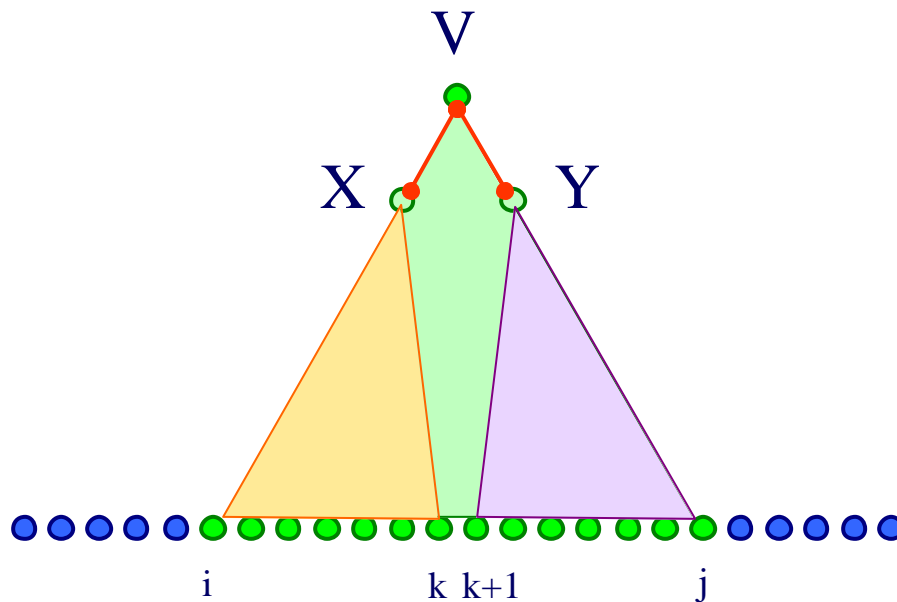
$D \rightarrow d$

The **Inside** Algorithm

To compute

$$a(i, j, V) = P(x_i \dots x_j \text{ produced by } V)$$

$$a(i, j, v) = \sum_X \sum_Y \sum_k a(i, k, X) a(k+1, j, Y) P(V \rightarrow XY)$$



Algorithm: Inside

Initialization:

For $i = 1$ to N , V a nonterminal,
 $a(i, i, V) = P(V \rightarrow x_i)$

Iteration:

For $i = 1$ to $N-1$
 For $j = i+1$ to N
 For V a nonterminal

$$a(i, j, V) = \sum_X \sum_Y \sum_k a(i, k, X) a(k+1, j, Y) P(V \rightarrow XY)$$

Termination:

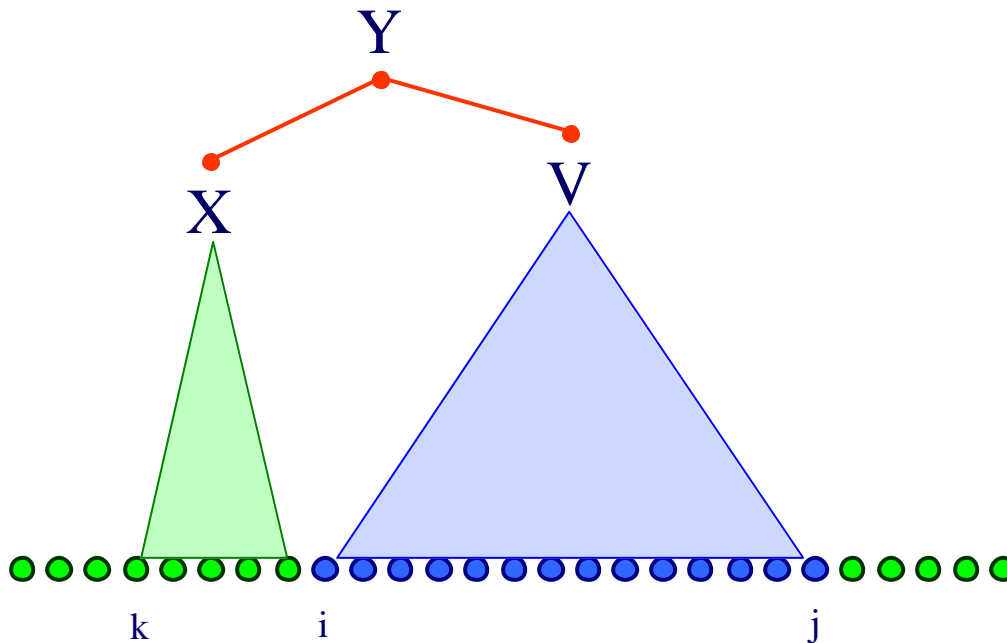
$P(x \mid \theta) = a(1, N, S)$

The Outside Algorithm

$b(i, j, V) = \text{Prob}(x_1 \dots x_{i-1}, x_{j+1} \dots x_N, \text{ where the "gap" is rooted at } V)$

Given that V is the right-hand-side nonterminal of a production,

$$b(i, j, V) = \sum_X \sum_Y \sum_{k < i} a(k, i-1, X) b(k, j, Y) P(Y \rightarrow XV)$$



Algorithm: Outside

Initialization:

$$b(1, N, S) = 1$$

For any other V , $b(1, N, V) = 0$

Iteration:

For $i = 1$ to $N-1$

For $j = N$ down to i

For V a nonterminal

$$b(i, j, V) = \sum_X \sum_Y \sum_{k < i} a(k, i-1, X) b(k, j, Y) P(Y \rightarrow XV) + \\ \sum_X \sum_Y \sum_{k < i} a(j+1, k, X) b(i, k, Y) P(Y \rightarrow VX)$$

Termination:

It is true for any i , that:

$$P(x \mid \theta) = \sum_X b(i, i, X) P(X \rightarrow x_i)$$

Learning for SCFGs

We can now estimate

$c(V)$ = expected number of times V is used in the parse of $x_1 \dots x_N$

$$c(V) = \frac{1}{P(x \mid \theta)} \sum_{1 \leq i \leq N} \sum_{i \leq j \leq N} a(i, j, V) b(i, j, v)$$

$$c(V \rightarrow XY) = \frac{1}{P(x \mid \theta)} \sum_{1 \leq i \leq N} \sum_{i < j \leq N} \sum_{i \leq k < j} b(i, j, V) a(i, k, X) a(k+1, j, Y) P(V \rightarrow XY)$$

Learning for SCFGs

Then, we can re-estimate the parameters with EM, by:

$$P^{\text{new}}(V \rightarrow XY) = \frac{c(V \rightarrow XY)}{c(V)}$$

$$P^{\text{new}}(V \rightarrow a) = \frac{c(V \rightarrow a)}{c(V)} = \frac{\sum_{i: x_i = a} b(i, i, V) P(V \rightarrow a)}{\sum_{1 \leq i \leq N} \sum_{i < j \leq N} a(i, j, V) b(i, j, V)}$$

Decoding: the CYK algorithm

Given $x = x_1 \dots x_N$, and a SCFG G ,

Find the most likely parse of x
(the most likely alignment of G to x)

Dynamic programming variable:

$\gamma(i, j, V)$: likelihood of the most likely parse of $x_i \dots x_j$,
 rooted at nonterminal V

Then,

$\gamma(1, N, S)$: likelihood of the most likely parse of x by the grammar

The CYK algorithm (Cocke-Younger-Kasami)

Initialization:

For $i = 1$ to N , any nonterminal V ,
 $\gamma(i, i, V) = \log P(V \rightarrow x_i)$

Iteration:

For $i = 1$ to $N-1$
For $j = i+1$ to N
For any nonterminal V ,

$$\gamma(i, j, V) = \max_X \max_Y \max_{i \leq k < j} \gamma(i, k, X) + \gamma(k+1, j, Y) + \log P(V \rightarrow XY)$$

Termination:

$$\log P(x \mid \theta, \pi^*) = \gamma(1, N, S)$$

Where π^* is the optimal parse tree (if traced back appropriately from above)

Summary: SCFG and HMM algorithms

<u>GOAL</u>	<u>HMM algorithm</u>	<u>SCFG algorithm</u>
Optimal parse	Viterbi	CYK
Estimation	Forward Backward	Inside Outside
Learning	EM: Fw/Bck	EM: Ins/Outs
Memory Complexity	$O(N K)$	$O(N^2 K)$
Time Complexity	$O(N K^2)$	$O(N^3 K^3)$

Where K : # of states in the HMM
of nonterminals in the SCFG

A SCFG for predicting RNA structure

$$\begin{aligned} S &\rightarrow a S \mid c S \mid g S \mid u S \mid \epsilon \\ &\rightarrow S a \mid S c \mid S g \mid S u \\ &\rightarrow a S u \mid c S g \mid g S u \mid u S g \mid g S c \mid u S a \\ &\rightarrow S S \end{aligned}$$

Adjust the probability parameters to be the ones reflecting the relative strength/weakness of bonds, etc.

Note: this algorithm does not model loop size!

CYK for RNA folding

Can do faster than $O(N^3 K^3)$:

Initialization:

$$\gamma(i, i-1) = -\text{Infinity}$$

$$\gamma(i, i) = \log P(x_i S)$$

Iteration:

For $i = 1$ to $N-1$

For $j = i+1$ to N

$$\gamma(i, j) = \max \left\{ \begin{array}{l} \gamma(i+1, j-1) + \log P(x_i S x_j) \\ \gamma(i+1, j) + \log P(S x_i) \\ \gamma(i, j-1) + \log P(x_i S) \\ \max_{i < k < j} \gamma(i, k) + \gamma(k+1, j) + \log P(S S) \end{array} \right.$$

The Zuker algorithm – main ideas

Models energy of an RNA fold

1. Instead of base pairs, pairs of base pairs (more accurate)
2. Separate score for bulges
3. Separate score for different-size & composition loops
4. Separate score for interactions between stem & beginning of loop

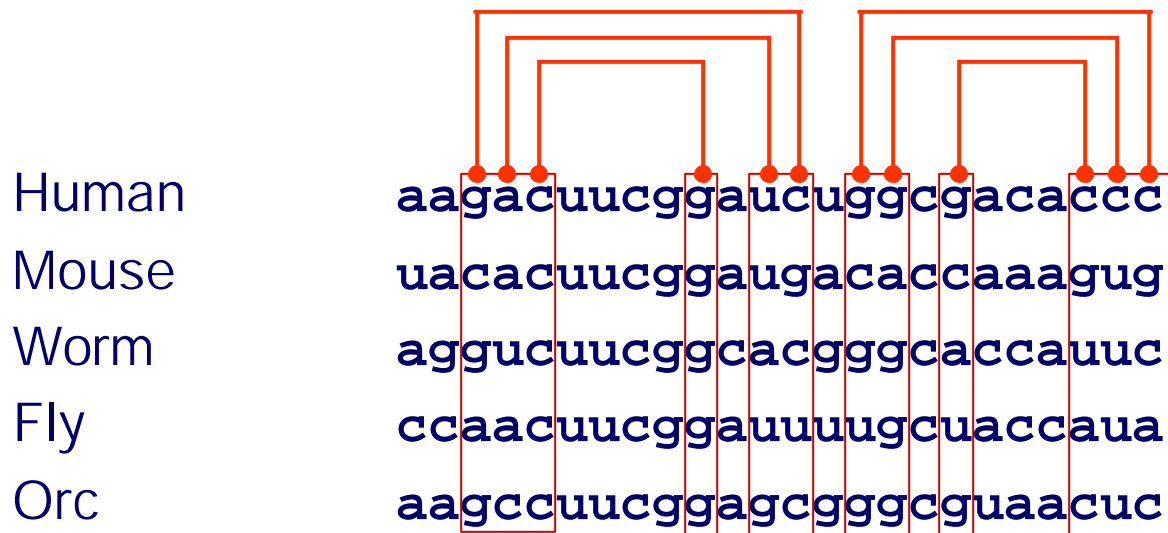
Can also do all that with a SCFG, and train it on real data

Methods for inferring RNA fold

- Experimental:
 - Crystallography
 - NMR
- Computational
 - Fold prediction (Nussinov, Zuker, SCFGs)
 - Multiple Alignment

Multiple alignment and RNA folding

Given K homologous aligned RNA sequences:



If i^{th} and j^{th} positions are always base paired and covary, then they are likely to be paired

Mutual information

$$M_{ij} = \sum_{a,b \in \{a,c,g,u\}} f_{ab}(i,j) \log_2 \frac{f_{ab}(i,j)}{f_a(i) f_b(j)}$$

Where $f_{ab}(i,j)$ is the # of times the pair a, b are in positions i, j

Given a multiple alignment, can infer structure that maximizes the sum of mutual information, by DP

In practice:

1. Get multiple alignment
2. Find covarying bases – deduce structure
3. Improve multiple alignment (by hand)
4. Go to 2

A manual EM process!!

Current state, future work

- The Zuker folding algorithm can predict good folded structures
- To detect RNAs in a genome
 - Can ask whether a given sequence folds well – not very reliable
- For tRNAs (small, typically ~60 nt; well-conserved structure)
Covariance Model of tRNA (like a SCFG) detects them well
- Difficult to make efficient probabilistic models of larger RNAs
- Not known how to efficiently do folding and multiple alignment simultaneously