Genome annotation – we discussed so far:

protein-coding genes

CpG islands - promoters

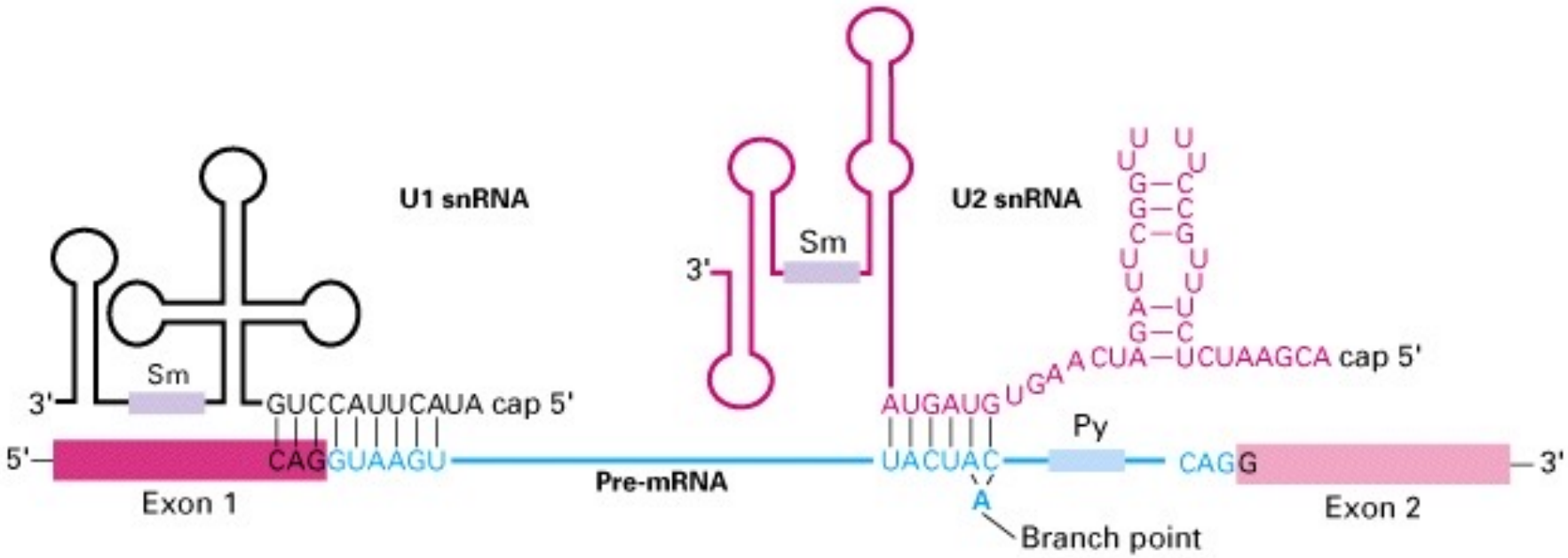
transcription regulatory elements

Genome annotation: non-coding RNAs

Classes of functional RNAs

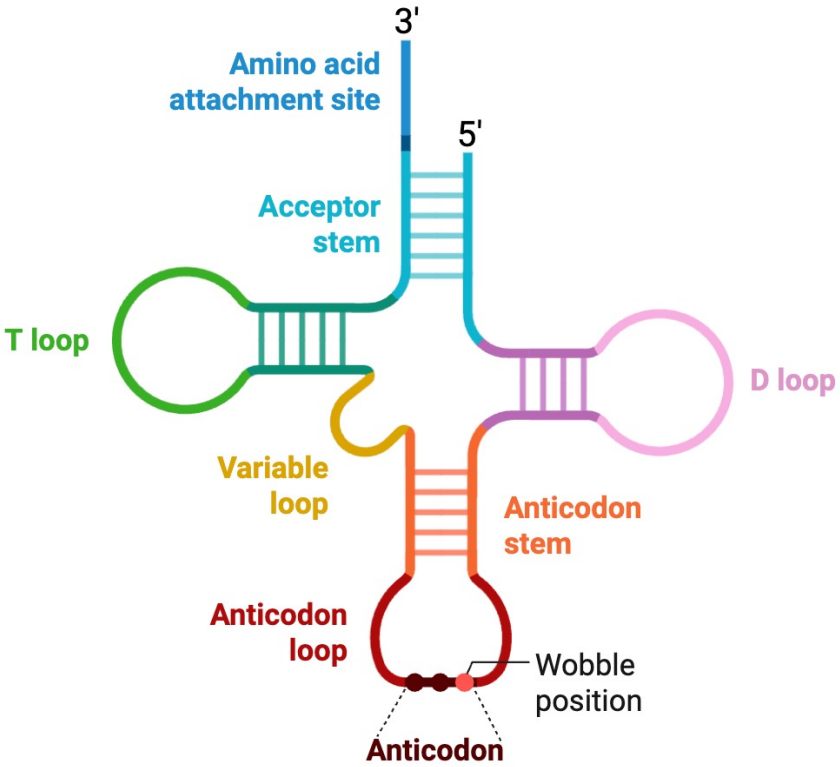
- rRNA - protein synthesis
- tRNA - transport of amino acids for protein synthesis
- snRNA - spliceosome
- snoRNA - rRNA methylation and pseudouridylation
- RNase P - removal of 5’ sequence from tRNAs
- 7-SL RNA in the SRP - protein secretion pathway
- miRNA, siRNA - translation inhibition / mRNA degradation
- piRNA - transposon inactivation
- antisense RNAs (Xist) - X chromosome inactivation
- bacterial noncoding RNAs - quorum sensing in Vibrio, etc.
- lncRNAs – long non-coding RNAs (lincRNA – intergenic)

snRNA in mRNA splicing



tRNAs in translation

Transfer RNA (tRNA) Structure



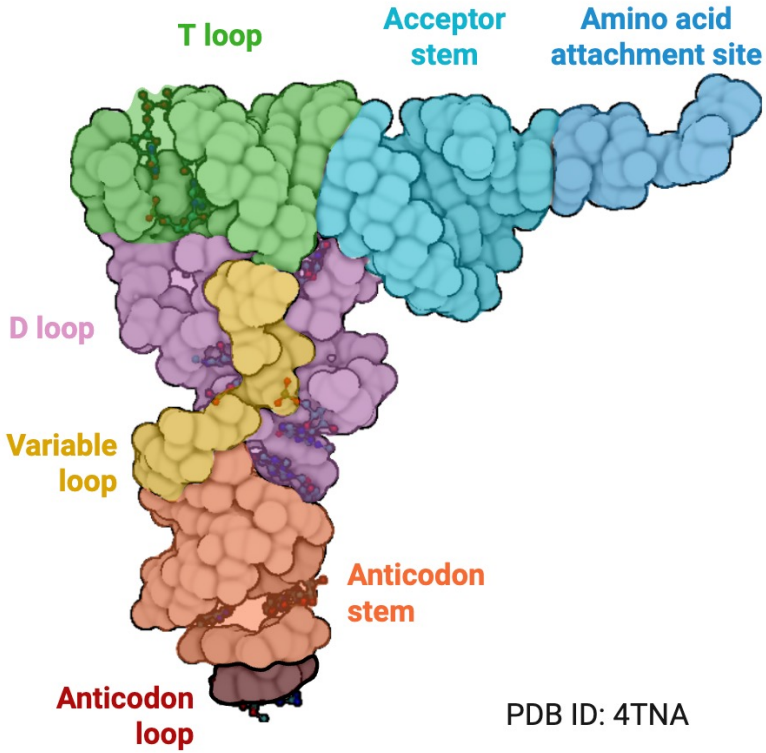
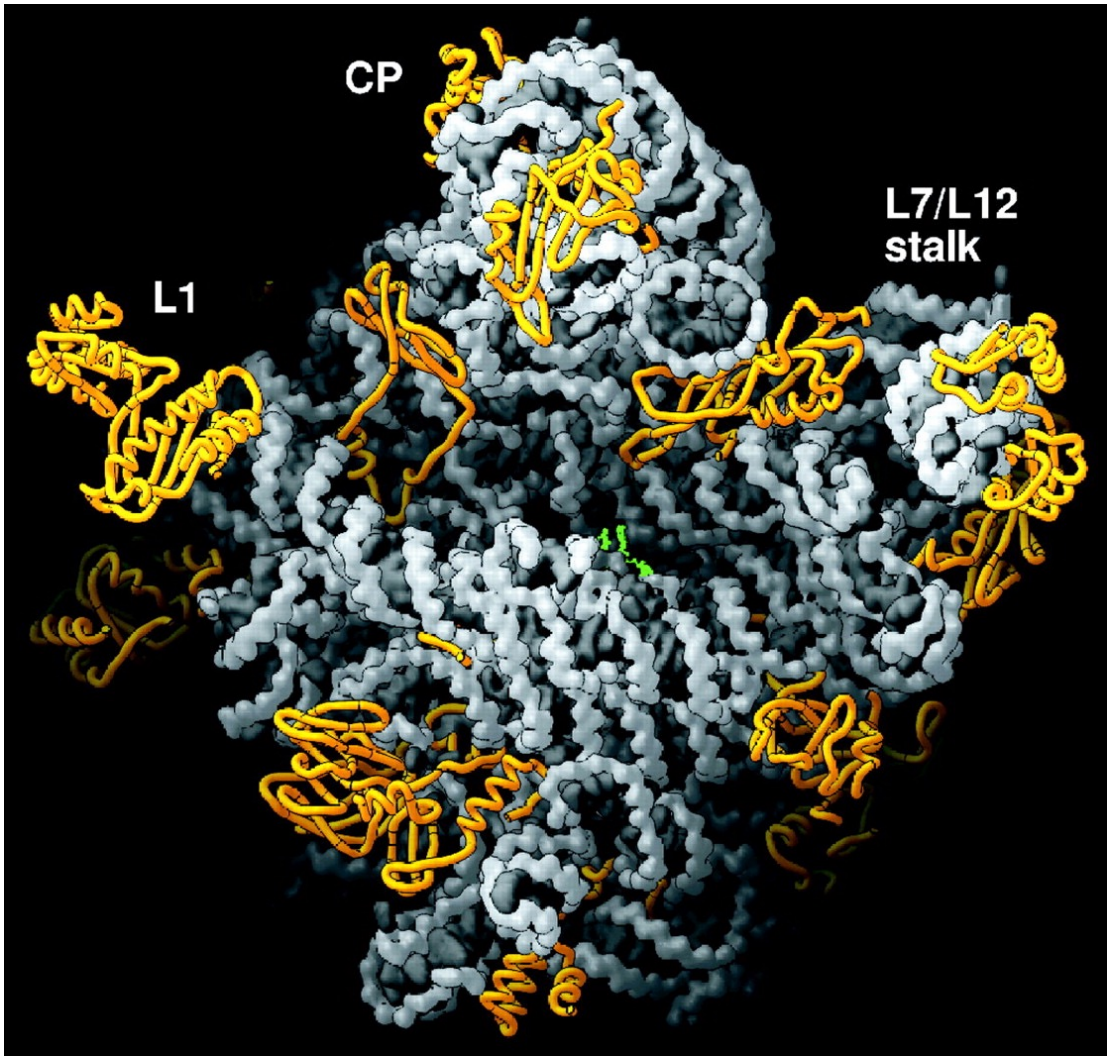


Figure from BioRender

rRNA in translation



RNA - gray
Peptide - gold
backbone

Large ribosomal subunit of Haloarcula marismortui - Ban et al. Science 289:905-920 (2000)

Some ncRNA databases

- general: ncRNA database
http://www.ncbi.nlm.nih.gov/RefSeq/

- microRNA repository

http://www.mirbase.org

- tRNA databases
http://lowelab.ucsc.edu/GtRNAdb/

- rRNA database
http://www.psb.ugent.be/rRNA/

Prediction of functional RNAs

Based on sequence homology to known RNA (e.g. rRNAs)

Based on structure homology (e.g. tRNAs)

Ab initio: based on the fact that functional RNAs have secondary structure that

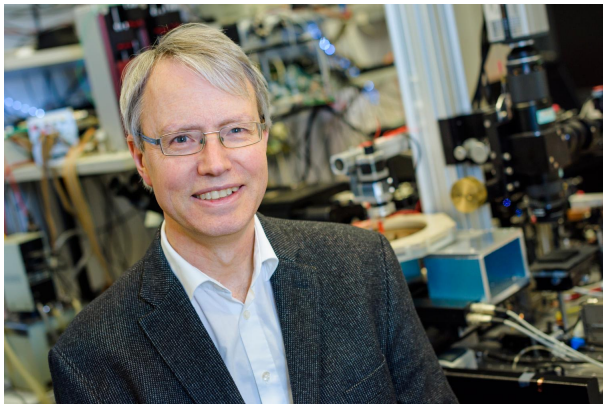
- Determines the function of the RNA
- To a reasonable approximation can be inferred from (relatively) simple rules

RNA structure prediction: an ‘old’ problem in computational biology

Manfred Eigen

Peter Schuster

Ivo Hofacker



John McCaskill

Why *predict* structures?

- To use as realistic toy models for studying evolutionary processes
- To generalize
- To expose model limitations of models and thereby improve our understanding RNA properties and functions

Continuity in Evolution: On the Nature of Transitions

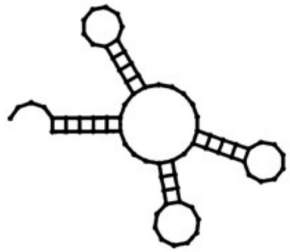
WALTER FONTANA AND PETER SCHUSTER Authors Info & Affiliations

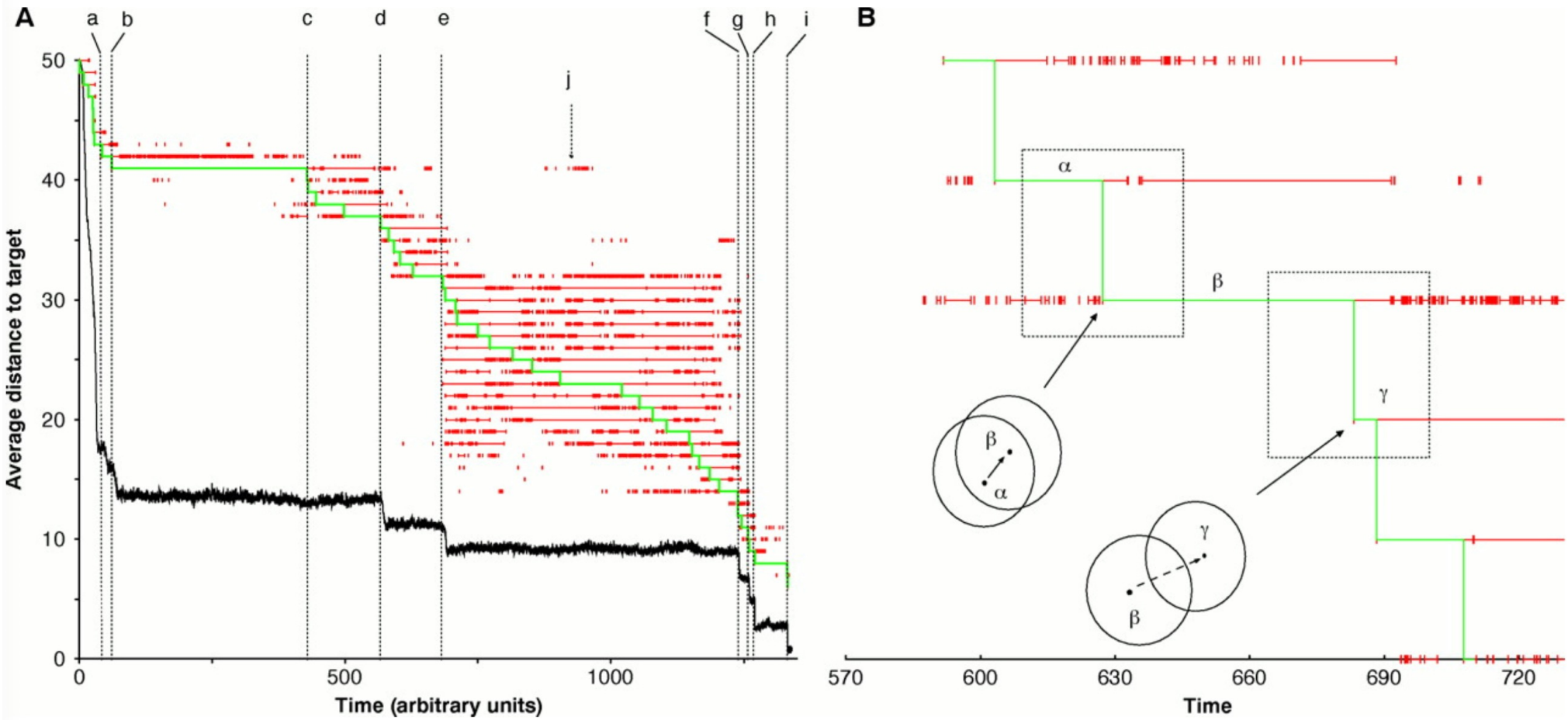
SCIENCE • 29 May 1998 • Vol 280, Issue 5368 • pp. 1451-1455 • DOI: 10.1126/science.280.5368.1451

Abstract

To distinguish continuous from discontinuous evolutionary change, a relation of nearness between phenotypes is needed. Such a relation is based on the probability of one phenotype being accessible from another through changes in the genotype. This nearness relation is exemplified by calculating the shape neighborhood of a transfer RNA secondary structure and provides a characterization of discontinuous shape transformations in RNA. The simulation of replicating and mutating RNA populations under selection shows that sudden adaptive progress coincides mostly, but not always, with discontinuous shape transformations. The nature of these transformations illuminates the key role of neutral genetic drift in their realization.

Target:

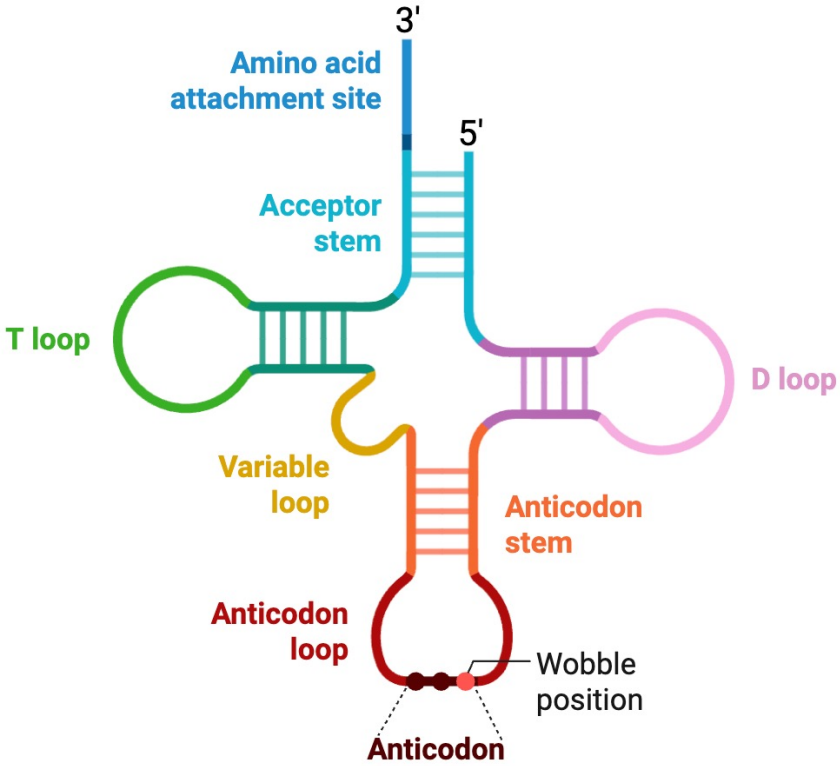


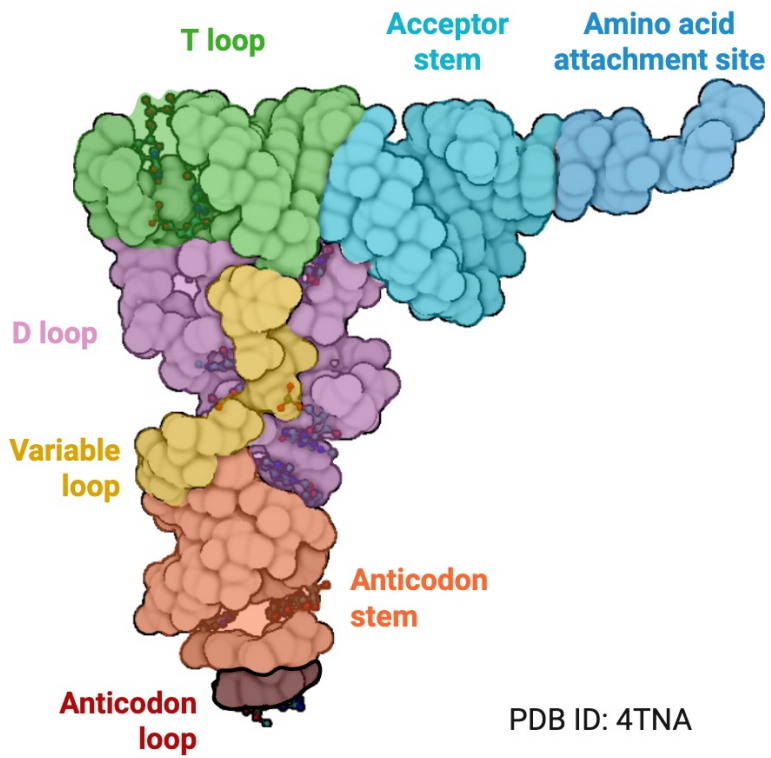

A
a
b
c
d
e
f
g
h
i
j
Average distance to target
Time (arbitrary units)
B
α
β
γ
Time

RNA structure

Tertiary structure:
Coordinates of the atoms

Primary structure:
Linear sequence of bases





Secondary structure:
Pattern of hydrogen bonding

Figure from BioRender

RNA secondary structure energetics

RNA secondary structure forms due to

Base pairing

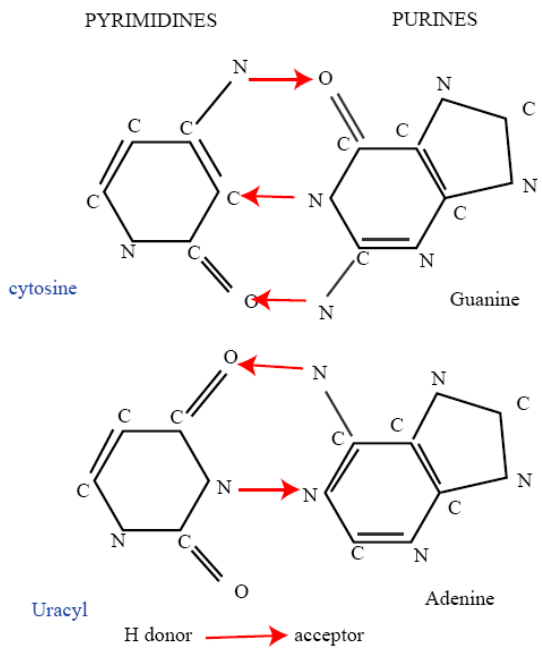
G-C ~ 3 kcal/mole

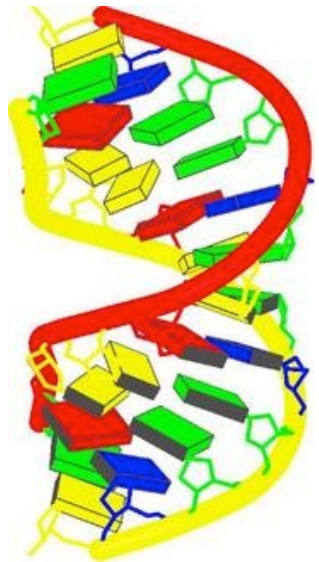
A-U ~ 2 kcal/mole

G-U ~ 1 kcal/mole

```
5'-GACCACGTGGTC-3'
   ||||||||||||
3'-CTGGTGCACCAG-5'
```

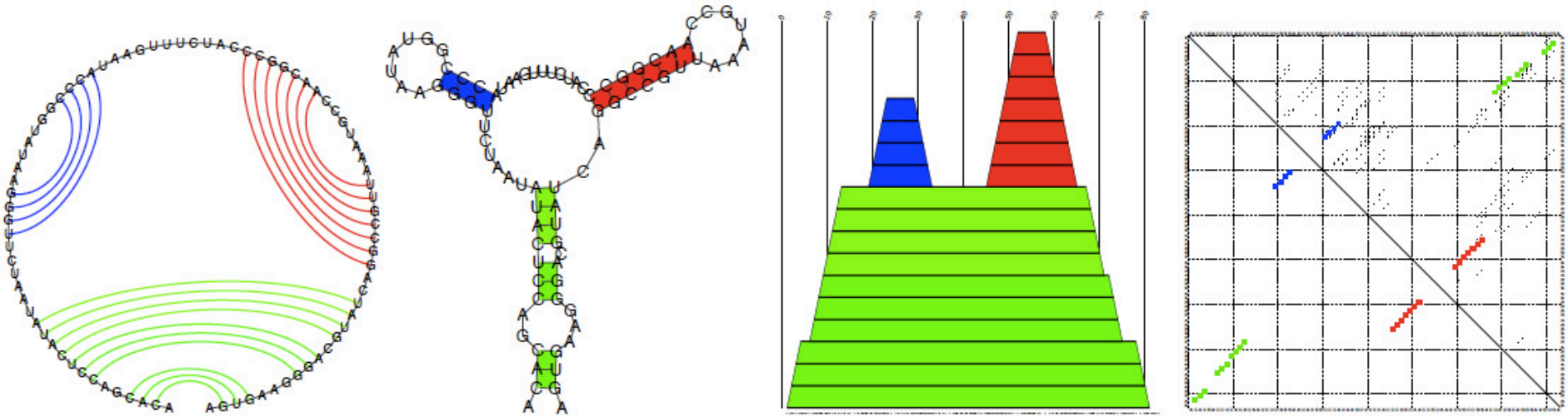
Base stacking





http://ndbserver.rutgers.edu

Representations of RNA secondary structure

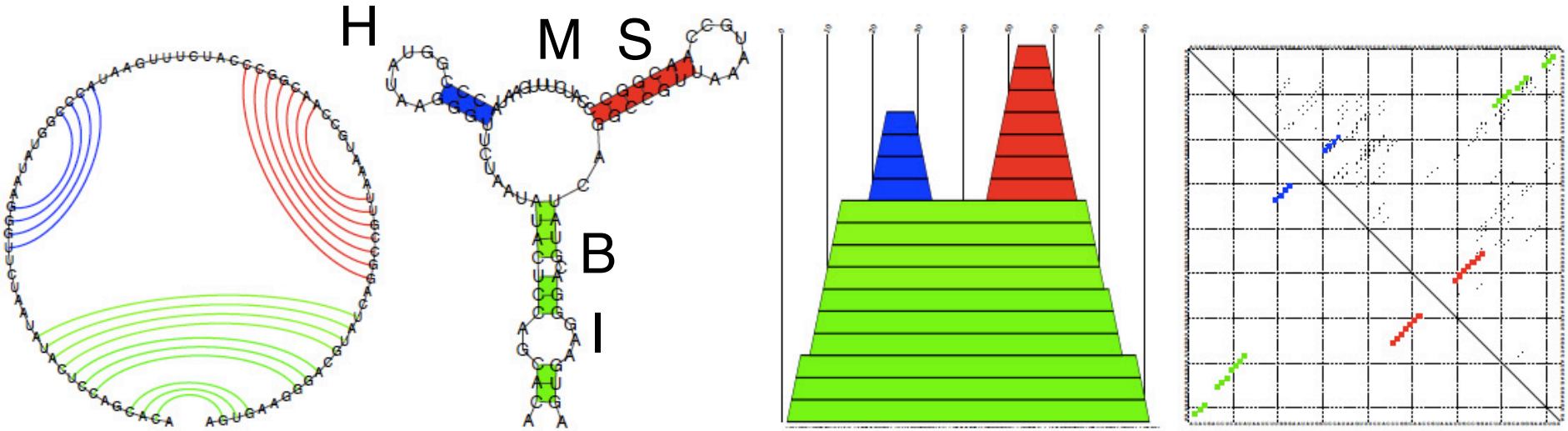


```
ACACGACCUCAUAUAAUCUUGGGAAUAUGGCCCAUAAGUUUCUACCCGGCAACCGUAAAUUGCCGGACUAUGCAGGGAAGUGA
.(((..(((((((.......((((.........))))............(((((((.......)))))))..))))).)))...)))).
```

Purine riboswitch (Rfam RF00167)

M - multi-loop
I - interior loop
B - bulge loop
H - hairpin loop
S - stack

Secondary structure glossary




```
ACACGACCUCAUAUAAUCUUGGGAAUAUGGCCCAUAAGUUUCUACCCGGCAACCGUAAAUUGCCGGACUAUGCAGGGAAGUGA
.(((..(((((((.....((((.......)))).............(((((((.......)))))))..)))).)))...)))).
```

Purine riboswitch (Rfam RF00167)

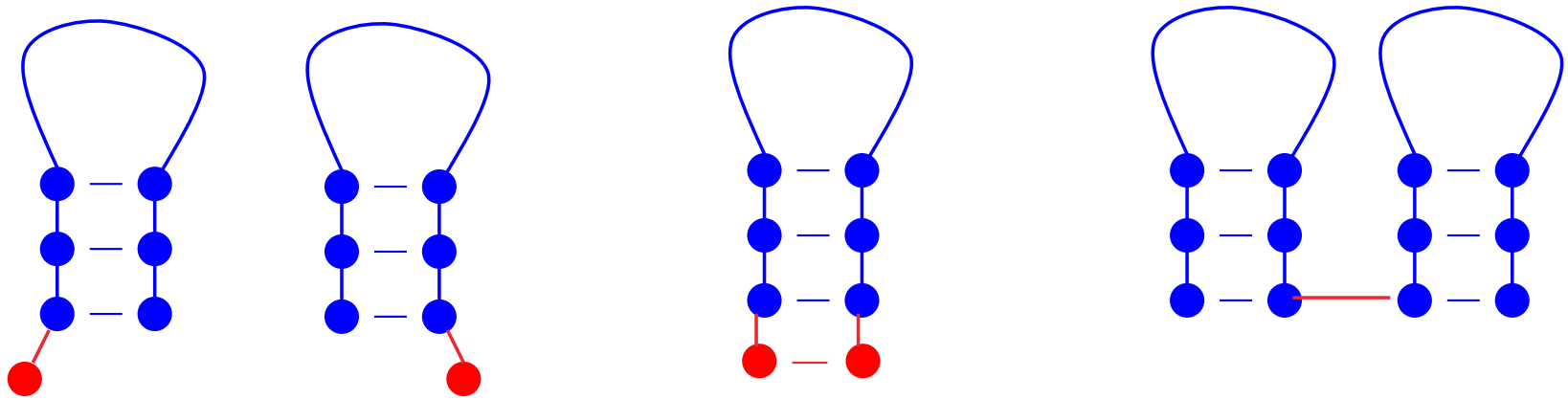
Prediction of secondary structure

1. Simple-minded: Maximization of the number of base pairs
2. General: Based on energetic constraints
3. Inference of ”energy” parameters with machine learning
4. Models of functional RNAs

Base pair maximization (Nussinov algorithm)

```
ACACGACCUCAUAUAAUCUUGGGAAUAUGGCCCAUAAGUUUCUACCCGGCAACCGUAAAUUGCCGGACUAUGCAGGGAAGUGA
.(((..(((((((......((((.......))))............((((((((.......)))))))..))))).)))...))).
```

Ways to “grow” a structure



Base pair maximization (Nussinov algorithm)

```
ACACGACCUCAUAUAAUCUUGGGAAUAUGGCCCAUAAGUUUCUACCCGGCAACCGUAAAUUGCCGGACUAUGCAGGGAAGUGA
.(((..(((((((......((((.......))))............(((((((.......)))))))..))))).)))...))).
```

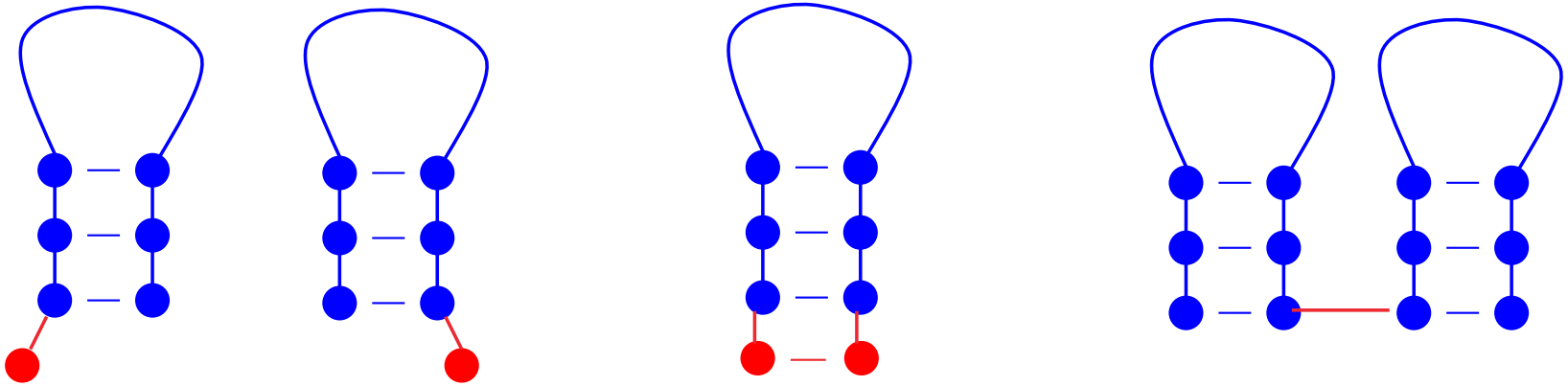
Define $\gamma(i,j)$ = maximum number of base pairs that can be formed for subsequence $s_i \ldots s_j$ and

$\delta(i,j)$ = score of pairing base s_i with base s_j.

Initialization: $\gamma(i,i) = 0, \forall\ i = 1 \ldots L$ and $\gamma(i-1,i) = 0, \forall\ i = 2 \ldots L$.

Base pair maximization (Nussinov algorithm)

```
ACACGACCUCAUAUAAUCUUGGGAAUAUGGCCCAUAAGUUUCUACCCGGCAACCGUAAAUUGCCGGACUAUGCAGGGAAGUGA
.(((..(((((((......((((.......))))............(((((((.......)))))))..))))).)))...))).
```



$\gamma(i+1,j)$ $\gamma(i,j-1)$ $\gamma(i+1,j-1)+\delta(i,j)$ $\gamma(i,k)+\gamma(k+1,j)$

Recursion: starting with all subsequences of length 2 to length *L*

$\gamma(i,j) = \max[\gamma(i+1,j),$ ← leaving base i unpaired

$\gamma(i,j-1),$ ← leaving base j unpaired

$\gamma(i+1,j-1)+\delta(i,j),$ ← pairing (s_i,s_j)

$\max_{i<k<j}[\gamma(i,k)+\gamma(k+1,j)].$ ←

combining two optimal substructures ($s_i..s_k$ and $s_{k+1}..s_j$)

Secondary structure prediction using Nussinov algorithm

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0										
C		0	0									
A			0	0								
G				0	0							
G					0	0						
A						0	0					
A							0	0				
C								0	0			
C									0	0		
A										0	0	
G											0	0
G												0

Secondary structure prediction using Nussinov algorithm

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0	0									
C		0	0									
A			0	0								
G				0	0							
G					0	0						
A						0	0					
A							0	0				
C								0	0			
C									0	0		
A										0	0	
G											0	0
G												0

$$\gamma(1,3) = \max(\gamma(2,3), \gamma(1,2), (\gamma(2,2) + \delta(1,3)))$$

Secondary structure prediction using Nussinov algorithm

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0	0									
C		0	0	1								
A			0	0								
G				0	0							
G					0	0						
A						0	0					
A							0	0				
C								0	0			
C									0	0		
A										0	0	
G											0	0
G												0

$$\gamma(2,4) = \max(\gamma(3,4), \gamma(2,3), (\gamma(3,3) + \delta(2,4)))$$

Secondary structure prediction using Nussinov algorithm

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0	0									
C		0	0	1								
A			0	0	0							
G				0	0	0						
G					0	0	0					
A						0	0	0				
A							0	0	0			
C								0	0	0		
C									0	0	1	
A										0	0	0
G											0	0
G												0

Secondary structure prediction using Nussinov algorithm

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0	0	1								
C		0	0	1								
A			0	0	0							
G				0	0	0						
G					0	0	0					
A						0	0	0				
A							0	0	0			
C								0	0	0		
C									0	0	1	
A										0	0	0
G											0	0
G												0

$$\gamma(1,4) = \max(\gamma(2,4), \gamma(1,3), (\gamma(2,3) + \delta(1,4)), (\gamma(1,2) + \gamma(3,4)))$$

Secondary structure prediction using Nussinov algorithm

Solution 1

```
   A A
 G - C
 G - C
A       A
 C - G
 C - G
```

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0	0	1	2	2	2	2	2	2	3	4
C		0	0	1	1	1	1	2	2	2	3	3
A			0	0	0	0	0	1	2	2	2	2
G				0	0	0	0	1	2	2	2	2
G					0	0	0	1	1	1	1	2
A						0	0	0	0	0	1	2
A							0	0	0	0	1	2
C								0	0	0	1	2
C									0	0	1	1
A										0	0	0
G											0	0
G												0

Secondary structure prediction using Nussinov algorithm

Solution 2

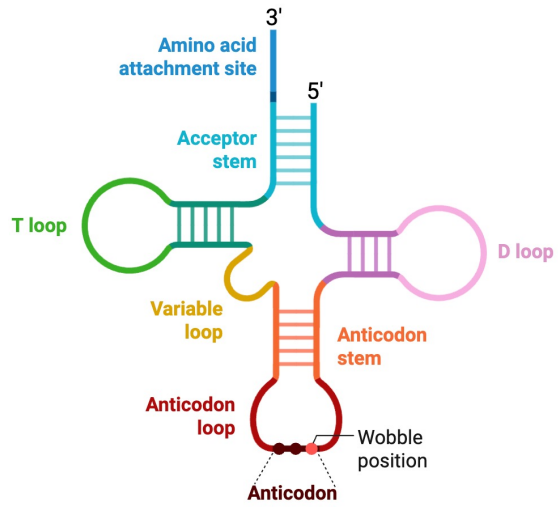
```
A     A
C-G   C-G
C-G   C-G
  A A
```

	C	C	A	G	G	A	A	C	C	A	G	G
C	0	0	0	1	2	2	2	2	2	2	3	4
C		0	0	1	1	1	1	2	2	2	3	3
A			0	0	0	0	0	1	2	2	2	2
G				0	0	0	0	1	2	2	2	2
G					0	0	0	1	1	1	1	2
A						0	0	0	0	0	1	2
A							0	0	0	0	1	2
C								0	0	0	1	2
C									0	0	1	1
A										0	0	0
G											0	0
G												0

Structure prediction based on folding free energy

5’ GCGAUUUAGCUCAGDDGGGGAGGCGCAGACUGAACAUCUGGAGGUCCUGUGUUCGAUCCACAGAAUUGCACCA 3’

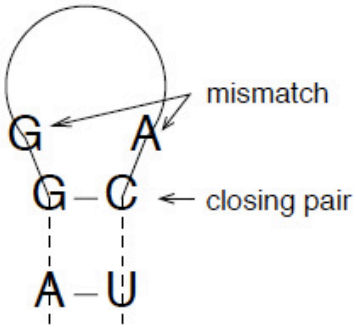
Folding free energy



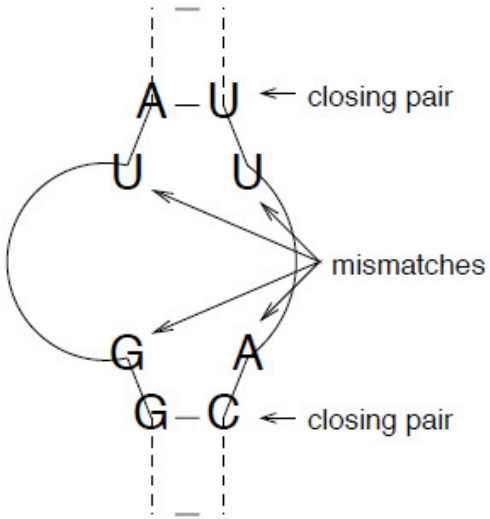
Loop decomposition of RNA structure

The energy of a structure is given by the sum of energies of smaller structural elements.

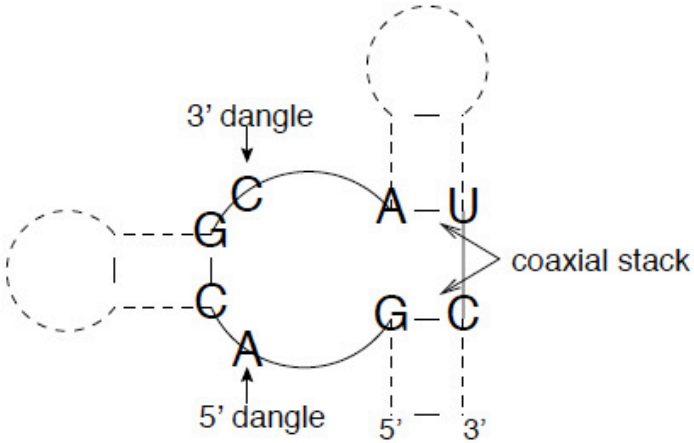
A practical decomposition has been in terms of “loops”, which are classified by the number of “closing pairs”:



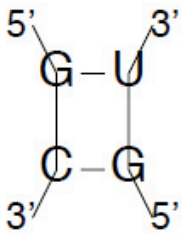
Hairpin Loop



Interior Loop



Multi Loop



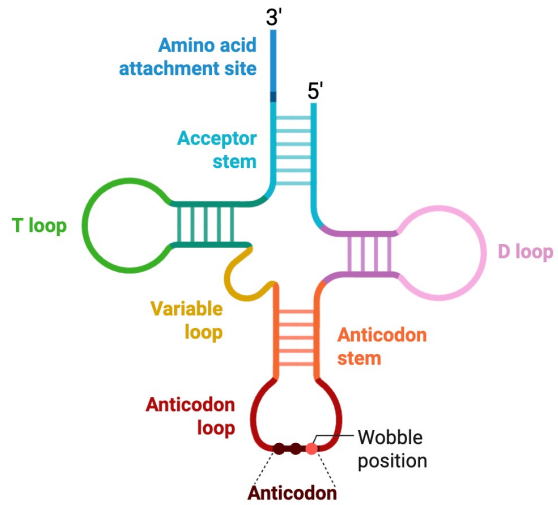
Stacked Pair

Structure prediction based on folding free energy

5’ GCGAUUUAGCUCAGDDGGGGAGGCGCAGACUGAACAUCUGGAGGUCCUGUGUUCGAUCCACAGAAUUGCACCA 3’

$$\Delta G = \Delta G(\text{exterior loop}) + \Delta G(\text{stacked pairs}) + \Delta G(\text{hairpin loops}) + \Delta G(\text{internal loops}) + \Delta G(\text{bulged loops}) + \Delta G(\text{multiloops}).$$

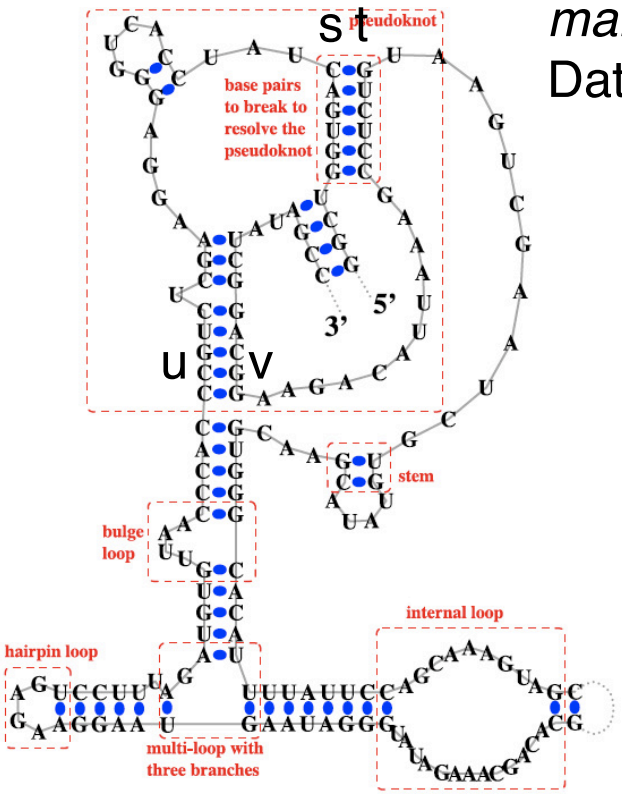
Folding free energy



Structure prediction based on folding free energy

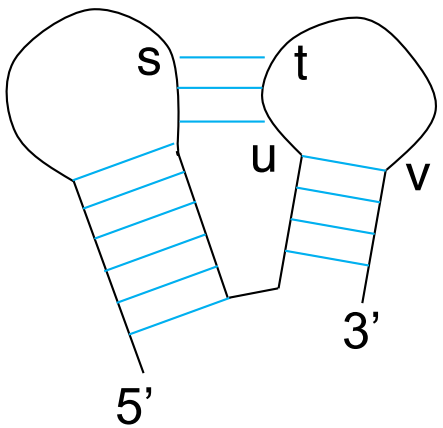
Efficient algorithms available if the structure does not have pseudoknots

A pseudoknotted secondary structure on an RNA sequence is a secondary structure in which there exist at least two base pairs {s, t} and {u, v}, for which s < u < t < v.



Structure of RNase P of *Methanococcus marapaludis* from the RNase P Database.

Such structures cannot be decomposed into disjoint secondary substructures with additive energy contribution.

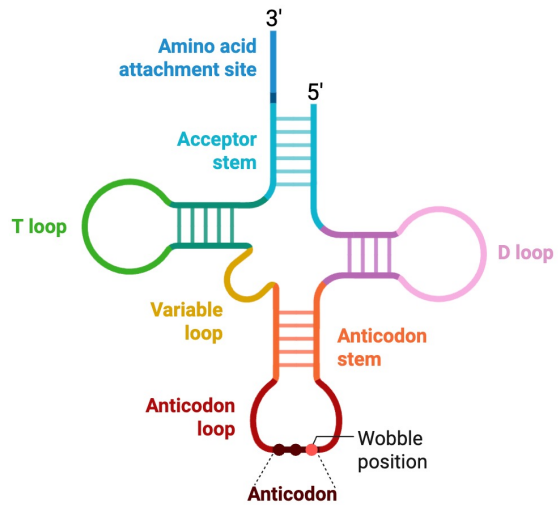


Structure prediction based on folding free energy

5' GCGAUUUAGCUCAGDDGGGGAGGCGCAGACUGAACAUCUGGAGGUCCUGUGUUCGAUCCACAGAAUUGCACCA 3'

$$\Delta G = \Delta G(\text{exterior loop}) + \Delta G(\text{stacked pairs}) + \Delta G(\text{hairpin loops}) + \Delta G(\text{internal loops}) + \Delta G(\text{bulged loops}) + \Delta G(\text{multiloops}).$$

Folding free energy

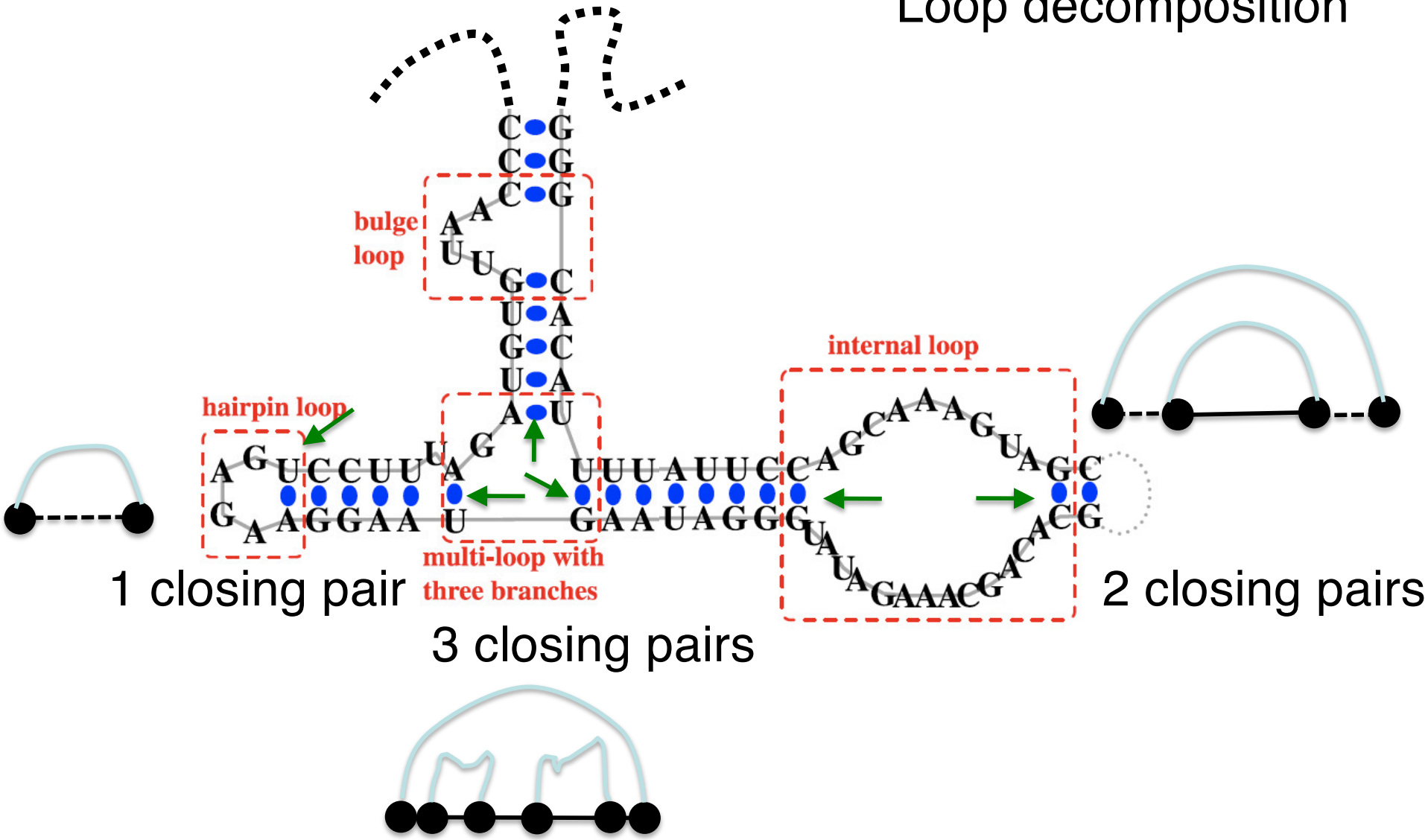


The energy of a structure relates to its frequency in the ensemble of structures that a given molecule can assume through the Bolzmann distribution: $f_i = \frac{exp\left(\frac{-E_i}{k_B T}\right)}{\sum_j exp\left(\frac{-E_j}{k_B T}\right)}$, where E_i is the energy of the structure f_i is its frequency, k_B is the Boltzmann constant and T is the temperature. The denominator is known as the partition function (typically called Z).

We could use the partition function formalism to compute posterior probabilities of base-pairs, or we could compute minima instead of sums to get the minimum free energy structure.

Computing the partition function

Loop decomposition



Computing the partition function

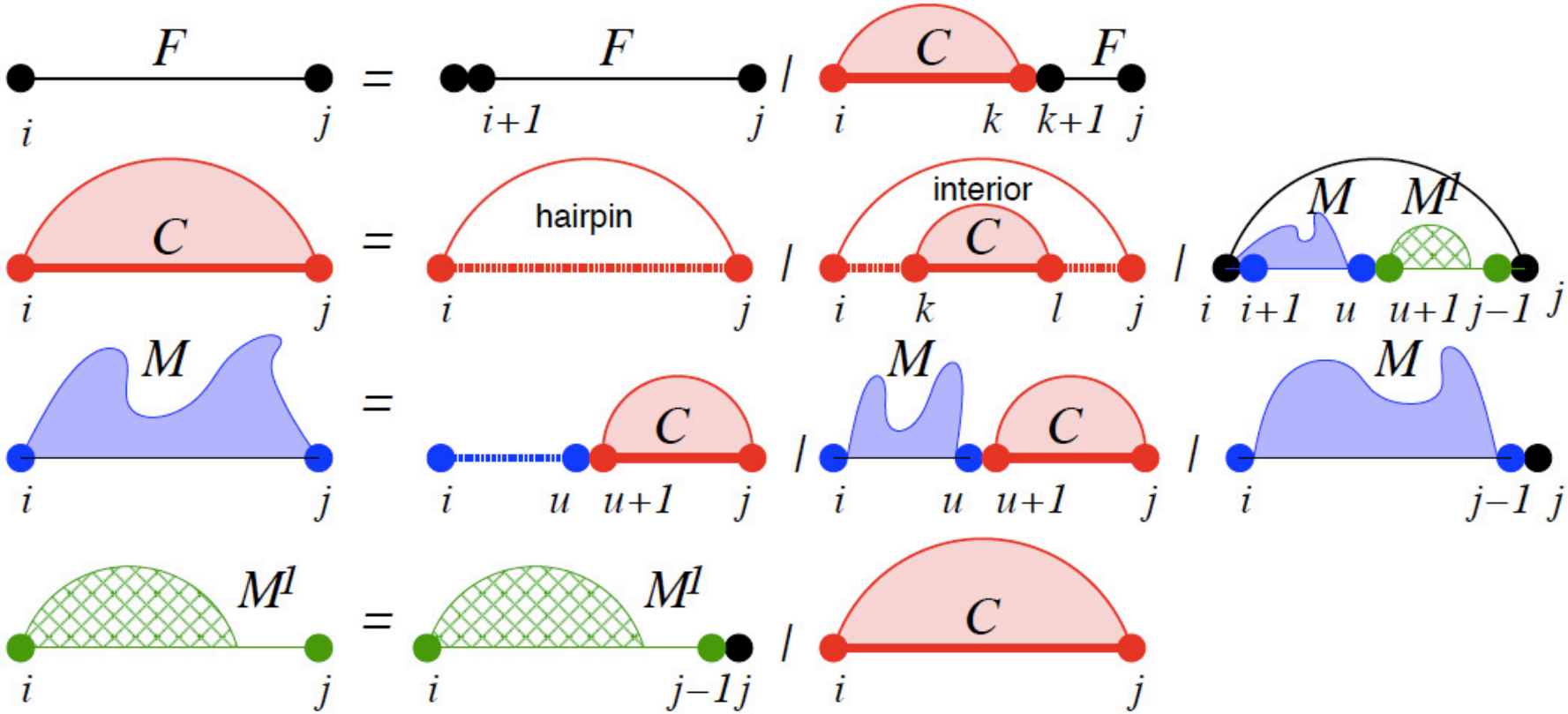
Loop decomposition in the Vienna RNA package

F_{ij} = Structure on the subsequence $x[i,j]$.

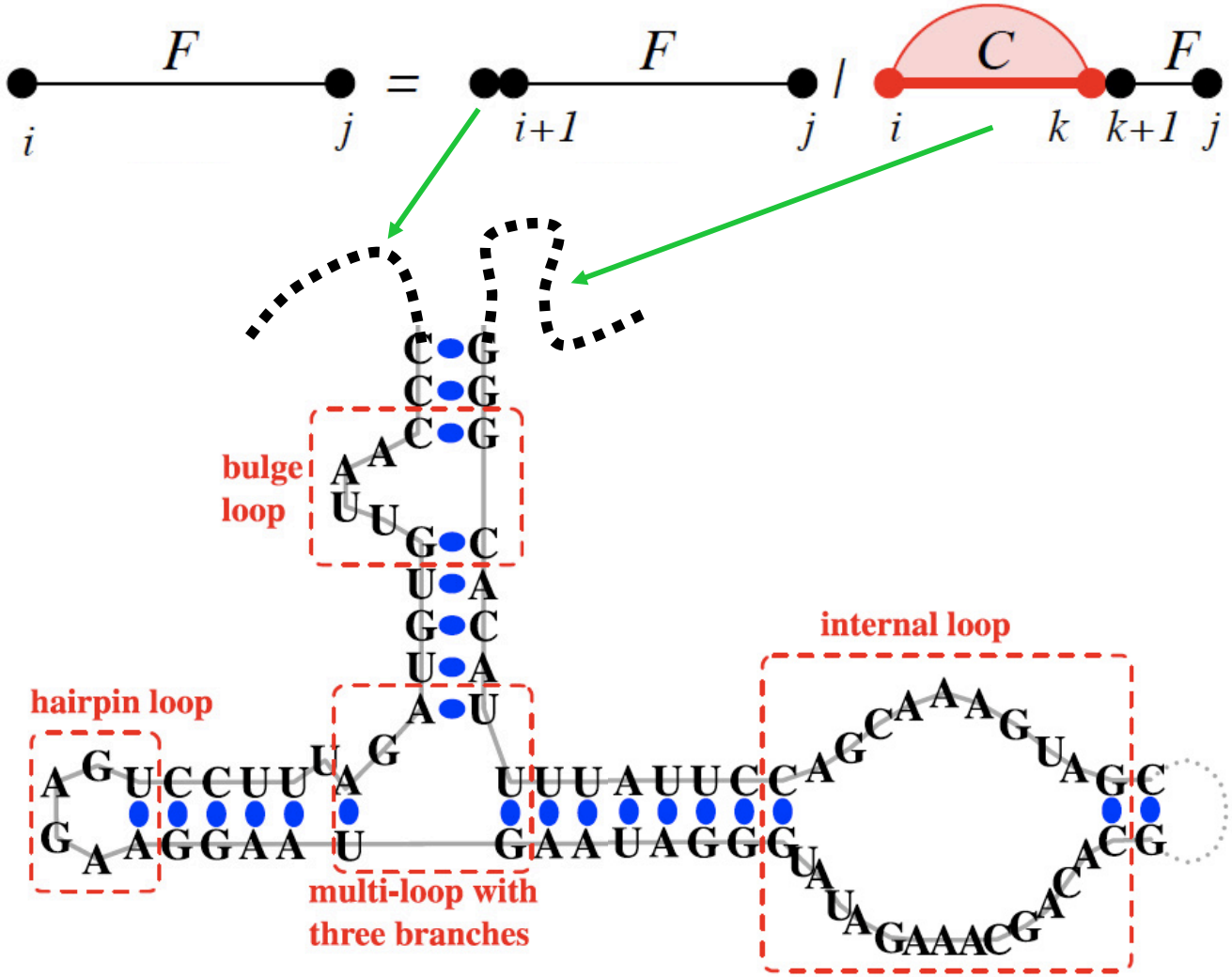
C_{ij} = Structure on the subsequence $x[i,j]$, subject to the constraint that i forms a pair with j.

M_{ij} = Structure on the subsequence $x[i,j]$, subject to the constraint that x[i,j] is part of a multiloop and has at least one component (subsequence enclosed by a base pair).

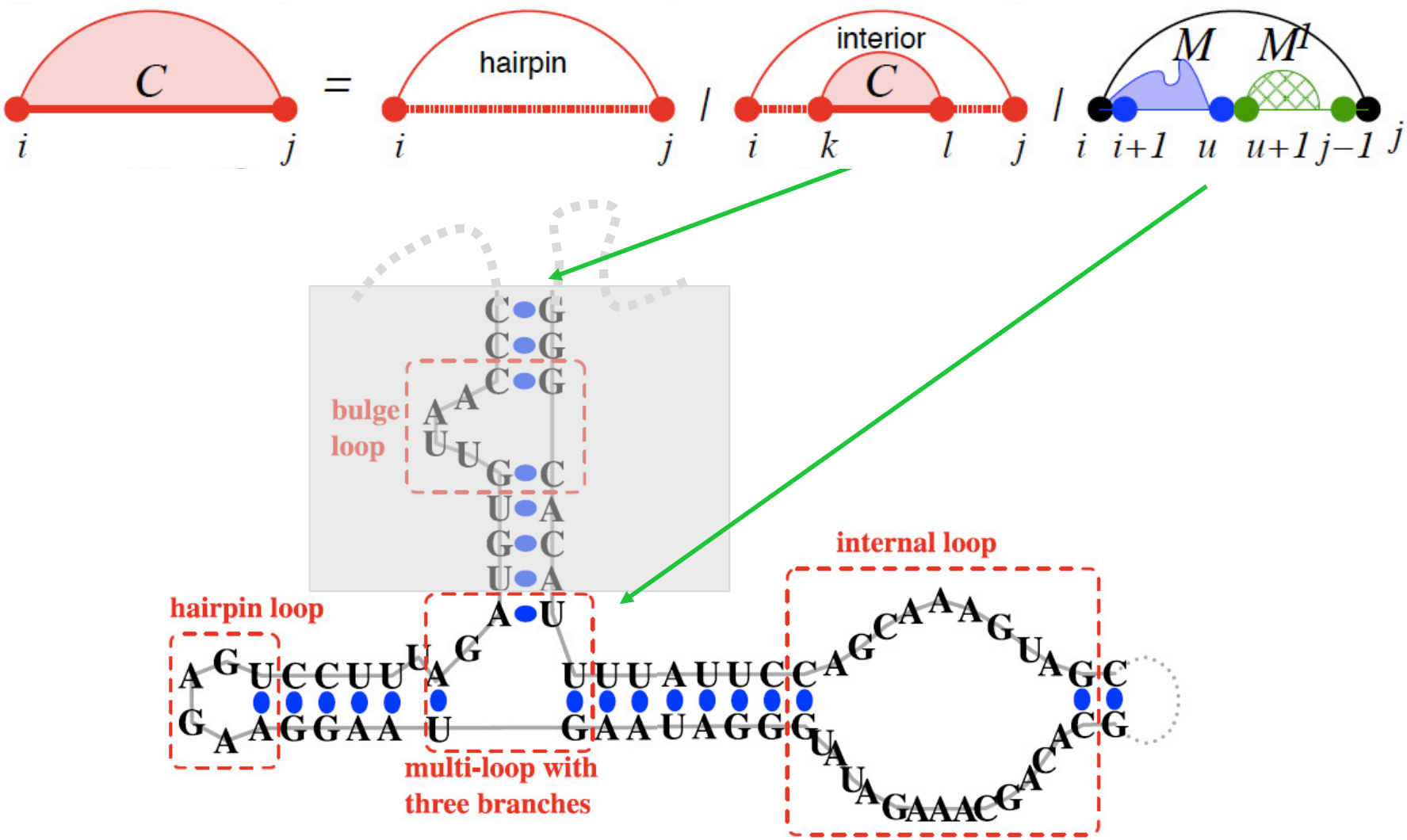
M^1_{ij} = Structure on the subsequence $x[i,j]$, subject to the constraint that $x[i,j]$ is part of a multiloop and has exactly one component, which has the closing pair (i,h) with $i<h \leq j$.



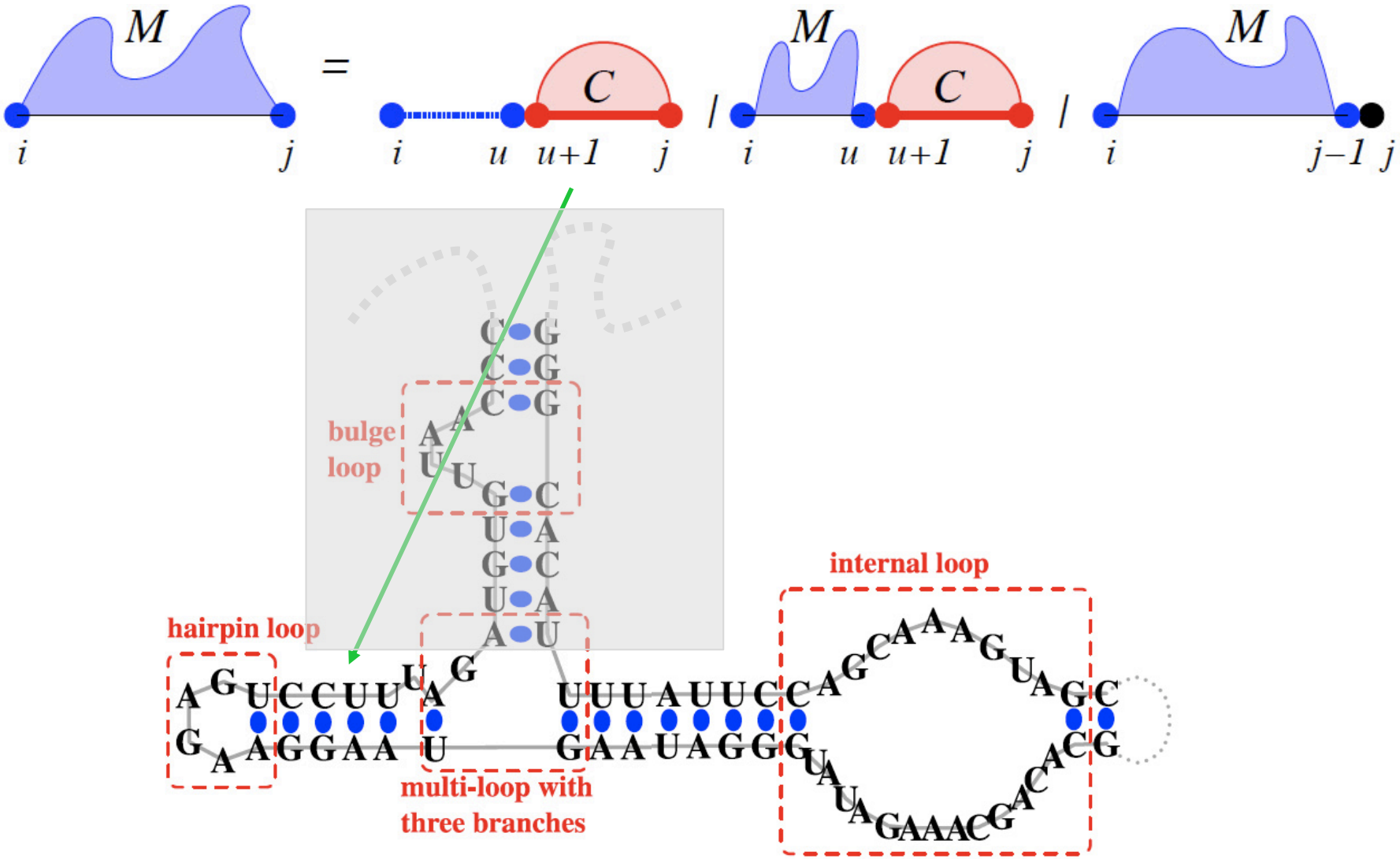
Computing the partition function



Computing the partition function

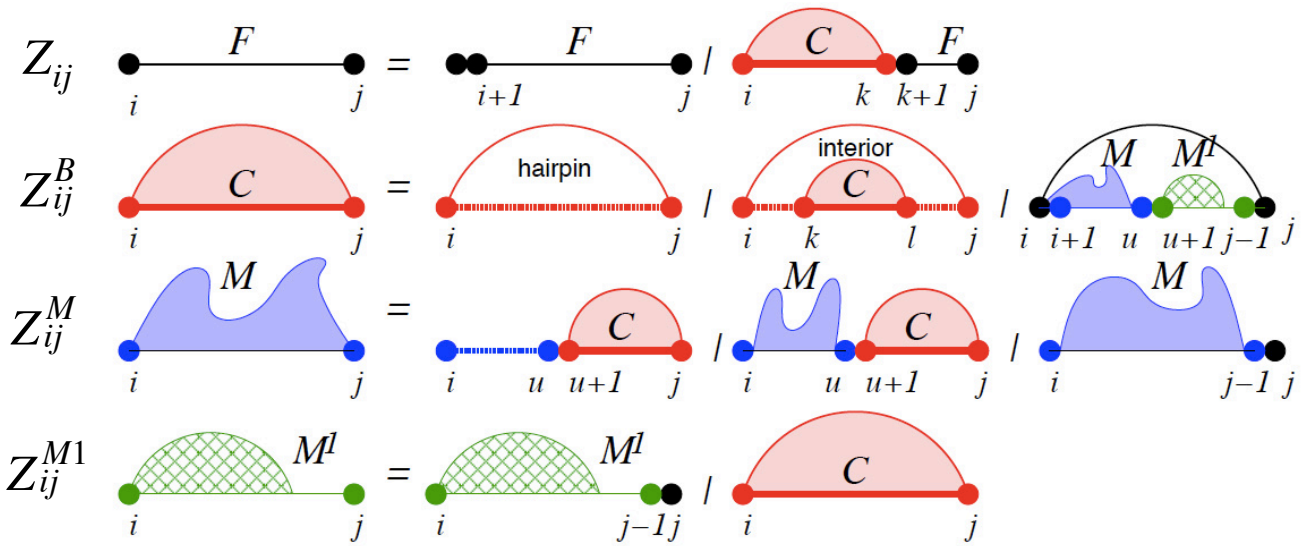


Computing the partition function



Computing the partition function

Loop decomposition in the Vienna RNA package



$H(i,j)$ – score for a hairpin loop of size $i - j - 1$

$I(i,j;k,l)$ – score for an internal loop enclosed by (i,j) and (k,l) base pairs

a - score for a multi-loop

b - score for one branch of a multi-loop

c - score for a unpaired base

Recall

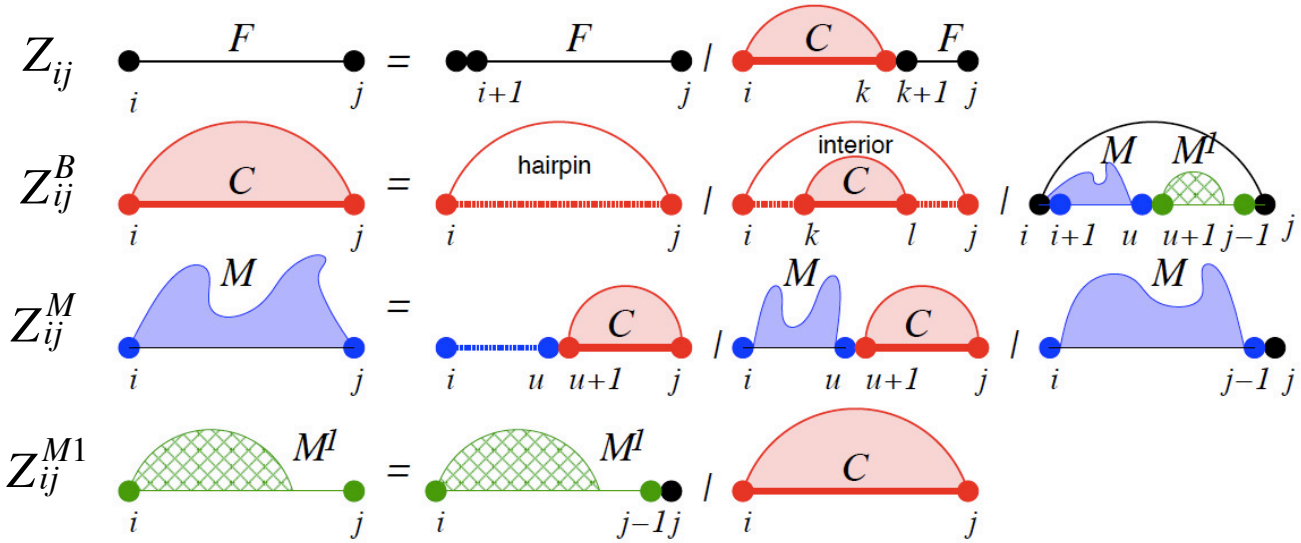
$$f_i = \frac{exp\left(\frac{-E_i}{k_B T}\right)}{\sum_j exp\left(\frac{-E_j}{k_B T}\right)}$$

$Z \sim$ probability

$H, I, etc. \sim$ energy

Computing the partition function

Loop decomposition in the Vienna RNA package



$$Z_{ij} = Z_{i+1,j} + \sum_{i<k\leq j} Z^B_{ik} Z_{k+1,j}$$

$$Z^B_{ij} = e^{-\beta H(i,j)} + \sum_{i<k<l<j} Z^B_{kl} e^{-\beta H(i,j;k,l)} + \sum_{i<u<j} Z^M_{i+1,u} Z^{M1}_{u+1,j} e^{-\beta a}$$

$$Z^M_{ij} = \sum_{i<u<j} e^{-\beta(u-i+1)c}\, Z^B_{u+1,j} + \sum_{i<u<j} Z^M_{i,u} Z^B_{u+1,j} e^{-\beta b} + Z^M_{i,j-1} e^{-\beta c}$$

$$Z^{M1}_{ij} = Z^{M1}_{i,j-1} e^{-\beta c} + Z^B_{ij} e^{-\beta b}$$

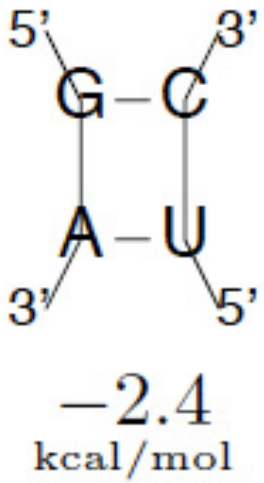
Boundary conditions: $Z_{ii} = 1, \;\; Z^B_{ii} = Z^{M1}_{ii} = Z^M_{ii} = 0$

Energy parameters

Widely used: Turner model, which is a nearest-neighbor model inferred from structure “melting” experiments

Free energies for stacked pairs (kcal/mol)

	CG	GC	GU	UG	AU	UA
CG	-2.4	-3.3	-2.1	-1.4	-2.1	-2.1
GC	-3.3	-3.4	-2.5	-1.5	-2.2	-2.4
GU	-2.1	-2.5	1.3	-0.5	-1.4	-1.3
UG	-1.4	-1.5	-0.5	0.3	-0.6	-1.0
AU	-2.1	-2.2	-1.4	-0.6	-1.1	-0.9
UA	-2.1	-2.4	-1.3	-1.0	-0.9	-1.3



Loop energies as a function of loop size

	1	2	3	4	5	6	7	8	9
hairpin	*	*	5.7	5.6	5.6	5.4	5.9	5.6	6.4
bulges	3.8	2.8	3.2	3.6	4.0	4.4	4.5	4.7	4.8
interior	*	*	*	1.7	1.8	2.0	2.2	2.3	2.4

Computing base pair probabilities

The probability to observe bases (k, l) paired in the ensemble of structures is given in terms of the total probability of structures containing the base pair $\sum_{(k,l)\in S} P(S)$

Contribution from structures in which (k, l) is not enclosed by other pairs

Contribution from structures with (k, l) closing a bulge or interior loop

$$P_{kl} = \frac{Z_{1,k-1} Z^B_{k,l} Z_{l+1,N}}{Z_{1N}} + \sum_{\{i,j|i<k<l<j\}} P_{ij} \frac{Z^B_{kl}}{Z^B_{ij}} e^{-\beta I(i,j;k,l)} +$$

$$\sum_{\{i,j|i<k<l<j\}} P_{ij} \frac{Z^B_{kl}}{Z^B_{ij}} e^{-\beta(a+b)} \left(e^{-\beta(k-i-1)c} Z^M_{l+1,j-1} + e^{-\beta(j-l-1)c} Z^M_{i+1,k-1} + Z^M_{i+1,k-1} Z^M_{k+1,j-1} \right)$$

Contribution from structures with (k, l) in a multiloop

universität wien

ViennaRNA Web Services

Institute for Theoretical Chemistry

Structure prediction | Folding Kinetics | Sequence Design | ncRNA Detection | Genome Wide Screening | Other

You are here: / RNA

Font size: A A A

Table of Contents

The ViennaRNA Web Services

This server provides programs, web services, and databases, related to our work on RNA secondary structures. For general information and other offerings from our group see the main TBI homepage.

To help us providing you with even better services please take the time to rate us at SurveyMonkey

Lots of tools at http://rna.tbi.univie.ac.at/

Our Web Services

Thermodynamic Structure Prediction

RNAfold Server

...predicts minimum free energy structures and base pair probabilities from single RNA or DNA sequences.

RNAprobing Server

...predicts minimum free energy structures and base pair probabilities from single RNAs using a guiding potential based on SHAPE reactivity probing data.

RNAalifold Server

...predicts *consensus* secondary structures from an alignment of several related RNA or DNA sequences. You need to upload an alignment.

RNAeval Server

...provides a detailed thermodynamic description of a

RNAcofold Server

...allows you to predict the secondary structure of a dimer.

RNAup Server

...allows you to predict the accessibility of a target region.

Getting energy parameters in the era of big data

Joint probability of the sequence *x* and structure *σ*:

$$P(x, \sigma) = \prod_{all\ structural\ elements\ in\ \sigma} p_{structural\ element}$$

Probability of the structure σ given sequence *x*:

$$P(\sigma|x) = \frac{P(x|\sigma)}{\sum_{\sigma'} P(x|\sigma')}$$

Denoting

p_i : probability of structural element *i* and $w_i = \ln(p_i)$

$F_i(x, \sigma)$: number of times structural element *i* has been used in structure σ

we can write the likelihood of the sequence *x* and structure σ as

$$P(x,\sigma) = \prod_{i=1}^{n} {p_i}^{F_i(x,\sigma)} = \exp\left(\ln\left(\prod_{i=1}^{n} {p_i}^{F_i(x,\sigma)}\right)\right)$$

$$= \exp\left(\sum_{i=1}^{n} F_i(x,\sigma)\ln(p_i)\right) = \exp\left(\mathbf{w}^{\mathrm{T}}\mathbf{F}(\mathrm{x},\sigma)\right)$$

and $P(\sigma|x) = \frac{\exp\left(\mathbf{w}^{\mathrm{T}}\mathbf{F}(\mathrm{x},\sigma)\right)}{\sum_{\sigma'}\exp\left(\mathbf{w}^{\mathrm{T}}\mathbf{F}(\mathrm{x},\sigma')\right)}$ Compare with $f_i = \frac{e^{-\frac{E_i}{k_B T}}}{\sum_j e^{-\frac{E_j}{k_B T}}}$

Inferring 'energy' parameters from a set of know structures

BIOINFORMATICS

Vol. 22 no. 14 2006, pages e90–e98
doi:10.1093/bioinformatics/btl246

CONTRAfold: RNA secondary structure prediction without physics-based models

Chuong B. Do[1,*], Daniel A. Woods[1] and Serafim Batzoglou[1]
[1]Computer Science Department, Stanford University, Stanford, CA 94305, USA

Find ***w*** that maximizes

$$\prod_{i=1}^{m} P\left(\boldsymbol{\sigma}^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{w}\right)$$

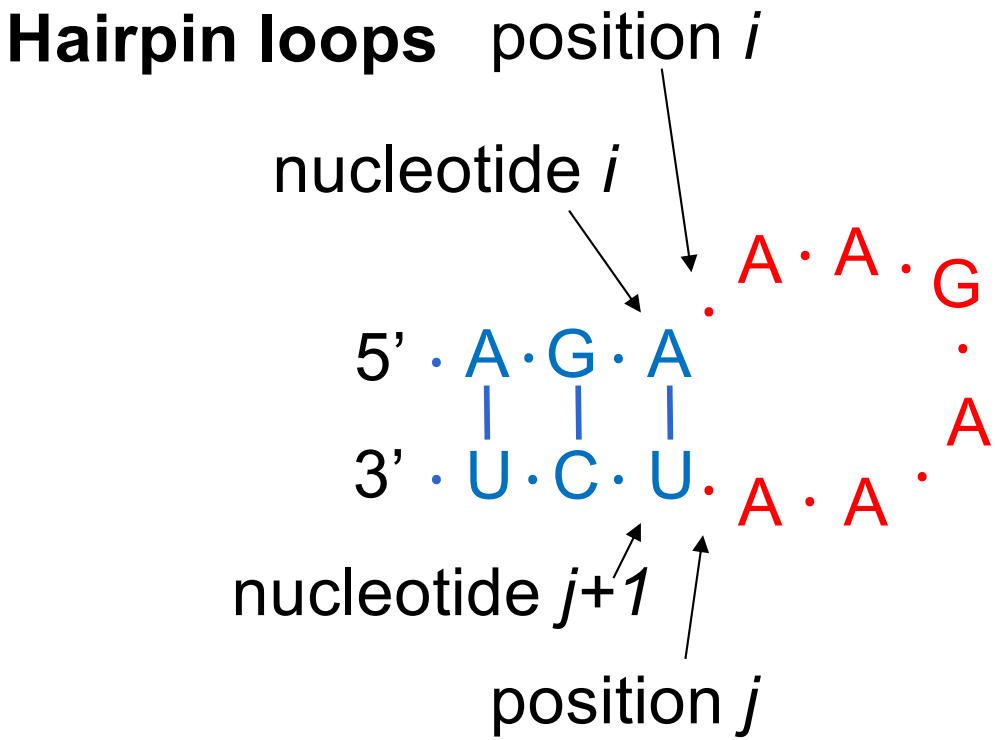
over sequences 1..*m* with validated structures

With $\mathrm{P}(\sigma|x) = \frac{\exp\left(\mathbf{w}^{\mathrm{T}}\mathbf{F}(\mathrm{x},\sigma)\right)}{\sum_{\sigma'} \exp\left(\mathbf{w}^{\mathrm{T}}\mathbf{F}(\mathrm{x},\sigma')\right)}$

Structural elements in CONTRAfold

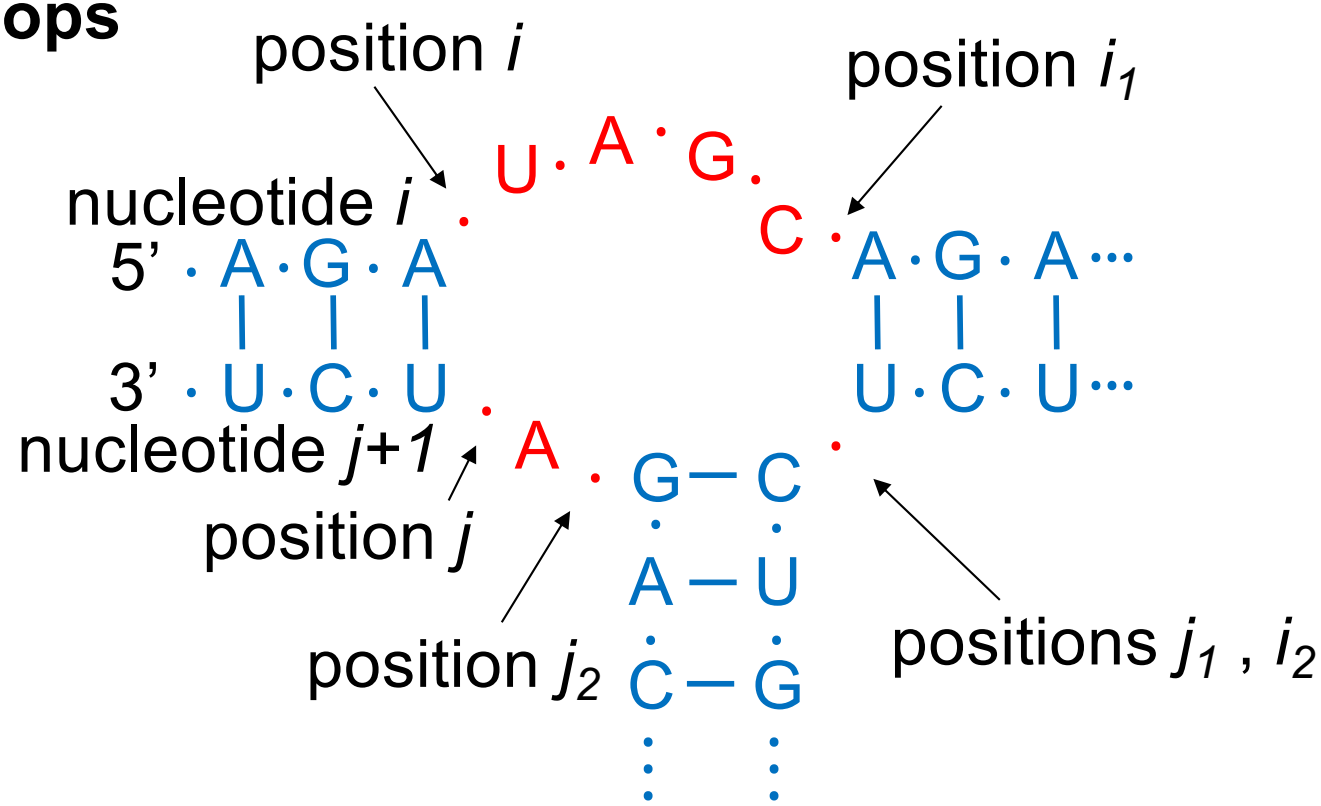
- Base pairs
- Helix closing base pairs
- Hairpin lengths
- Helix lengths
- Bulge lengths
- Internal loop lengths
- Internal loop asymmetry
- 2D table of internal loop scores
- Helix base pair stacking interactions
- Terminal mismatch interactions
- Single (dangling) base stacking
- Affine multi-branch loop scoring
- Free bases

Hairpin loops



$$w_{hairpin}(i,j) = w_{terminal\ mismatch}\big((x_i, x_{j+1}), x_{i+1}, x_j\big) + \begin{cases} w_{hairpin\ length}|j-i| \ if\ 0 \le j-i \le 30 \\ w_{loop\ base} + w_{loop\ multiplier} \cdot ln(j-i)\ if\ j-i > 30 \end{cases}$$

Multi-loops



Base score

Paired bases $w_{multi}(i, j, i_1, j_1, \ldots, i_m, j_m) = w_{multi\ base}$

$+ w_{multi\ paired} \cdot (m+1) + w_{multi\ unpaired} \cdot l$ Unpaired bases

$+ w_{stacking\ left}\left(\left(x_i, x_{j+1}\right), x_{i+1}\right) + w_{stacking\ right}\left(\left(x_i, x_{j+1}\right), x_j\right)$

$+ \sum_{k=1}^{m} w_{stacking\ left}\left(\left(x_{j_k}, x_{i_k+1}\right), x_{j_k+1}\right)$

$+ \sum_{k=1}^{m} w_{stacking\ right}\left(\left(x_{j_k}, x_{i_k+1}\right), x_{i_k}\right)$

Good correspondence of the learned model with what we know already

(a) Learned

$5' \longrightarrow 3'$

aX

uY

$3' \longleftarrow 5'$

		Y			
		a	c	g	u
X	a	0.48	0.38	0.34	-1.24
	c	0.27	0.33	-1.74	0.34
	g	0.34	-1.63	0.27	-0.74
	u	-1.26	0.32	-0.89	0.32

(b) Experimental

$5' \longrightarrow 3'$

aX

uY

$3' \longleftarrow 5'$

		Y			
		a	c	g	u
X	a	.	.	.	-0.90
	c	.	.	-2.20	.
	g	.	-2.10	.	-0.60
	u	-1.10	.	-1.40	.

Secondary structure models

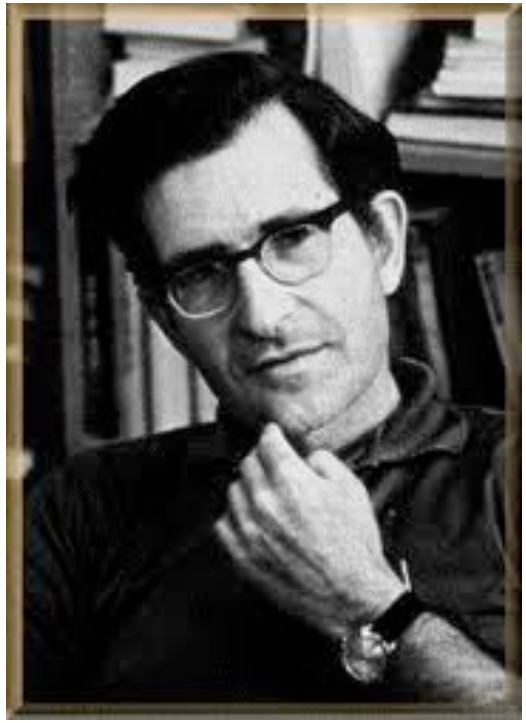
So far we looked at approaches that aim to predict the structure of an RNA molecule *ab initio*.

What if we already have a number of examples from a given family of RNA molecules (e.g. tRNAs) with a characteristic structure? How would we find additional examples of such RNAs in genomic sequences?

General approach: we construct a model from the known examples and then we search for additional sequences that fit that model.

Context-free grammars

Terminology coming from linguistics. With the motivation to determine what kind of language constructions (sentences) can be recognized by a machine, Noam Chomsky described in 1956 a hierarchy of formal grammars.



A formal grammar consists of:
- start symbol
- finite set of terminal symbols (elementary symbols that cannot be broken into smaller units), usually represented in lower-case
- finite set of non-terminal symbols, usually represented in upper-case
- finite set of production rules

Rule application: replacing the symbols that appear on the left-hand side of the rule with the symbols from the right-hand side of the rule.
Derivation: sequence of rule applications

Formal language: all strings consisting entirely of terminal symbols that can be reached by a derivation from the start symbol.

Formal languages: example

Consider the language that has the terminal symbols {(,)}, the non-terminal symbol S, and the rule $S \rightarrow \varepsilon | (S)$, where ε is the empty string. If we view these terminals as 5' and 3' members of a base pair, we have just described the language of all hairpins.

Example derivation:

$$
\begin{array}{cc}
S \rightarrow (S) & (S) \\
S \rightarrow (S) & ((S)) \\
S \rightarrow (S) & (((S))) \\
S \rightarrow (S) & ((((S)))) \\
S \rightarrow \varepsilon & ((((\varepsilon))))
\end{array}
\qquad\qquad
((()))
$$

Formal languages: example

What we usually want to find out in practice is whether a given string is in a given language.
E.g. is the string (((()))) part of the language with the following grammar

Terminal symbols $\{(,)\}$

Non-terminal symbol S

Production rules $S \rightarrow (S) \mid \varepsilon$

?

We answer this question generally, by writing an algorithm that describes how the computer would parse a valid string in the given language, applying the production rules. If at any point while reading the input string we cannot apply any of the grammar rules, the string can not be in the language. For the example above, the algorithm would look like this:

```
bool valid(s,i,j) {
    if(i > j)
            return True
    else if(i == j)
            return False
    else  if(s[i] == '(' && s[j] == ')')
            return valid(s,i+1,j-1)
    return False
}
```

```
( ( ( ( ) ) ) )
12345678
```

valid(s,1,8)

valid(s,2,7)

valid(s,3,6)

valid(s,4,5)

valid(s,5,4)

Chomsky hierarchy

In the order of decreasing restrictions imposed on the production rules, the Chomsky hierarchy of languages includes:

1. Regular grammars: only productions of the sort $S \rightarrow a|aS$ are allowed. This grammar will generate strings in the language from left to right (right to left equivalent is also a regular language).
2. Context-free grammars: allow productions of the type $S \rightarrow \alpha$, where α is an expression constructed from non-terminals and terminals.
3. Context-dependent grammars: allow productions of the sort $\alpha_1 S \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$.
4. Unrestricted grammars: any production of the sort $\alpha_1 S \alpha_2 \rightarrow \gamma$ is allowed.

Each type of grammar has a corresponding parser that can recognize sentences in that grammar:

Regular grammar - Finite state automaton
Context-free grammar - Pushdown automaton
Context-sensitive grammar - Linear bounded automaton
Unrestricted grammar - Turing machine

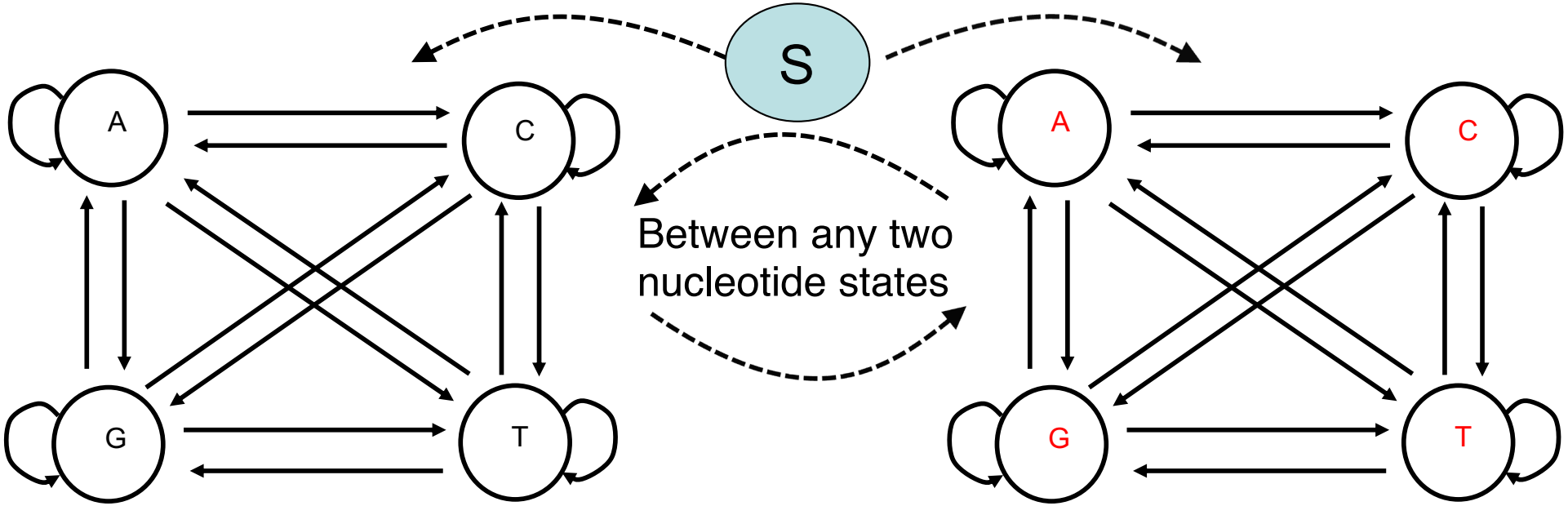
Finite state automata for representing sequences

Example: representing a genomic sequence as a sequence of CpG and non-CpG regions

$$S \rightarrow \Lambda \mid \Gamma$$

$$\Gamma \rightarrow A\Gamma \mid C\Gamma \mid G\Gamma \mid T\Gamma \mid A\Lambda \mid C\Lambda \mid G\Lambda \mid T\Lambda \mid \varepsilon$$

$$\Lambda \rightarrow A\Lambda \mid C\Lambda \mid G\Lambda \mid T\Lambda \mid A\Gamma \mid C\Gamma \mid G\Gamma \mid T\Gamma \mid \varepsilon$$



Markov Model for CpG islands– stochastic variant of the regular grammar.

Stochastic grammars

Every grammar has its stochastic counterpart.

Whereas with a non-stochastic grammar a string can either be generated through a derivation of not (in which case the string is not in the language), with a stochastic grammar, each string has an associated probability to be generated.

To obtain a stochastic variant of a grammar, one has to associate specific probabilities to all the productions of the grammar. The sum of probabilities of all the productions from a non-terminal must be 1.

Pushdown automata

Pushdown automata are the machines that process context-free grammars, just like the finite state automata are machines that process regular grammars.

PDAs are usually described in terms of
- an input tape
- a finite control and
- a stack, which is a string of symbols from some alphabet.

The machine has two types of moves:
- In the first type of moves, depending on
    - the input symbol
    - the symbol at the top of the stack, and
    - the state of the finite control

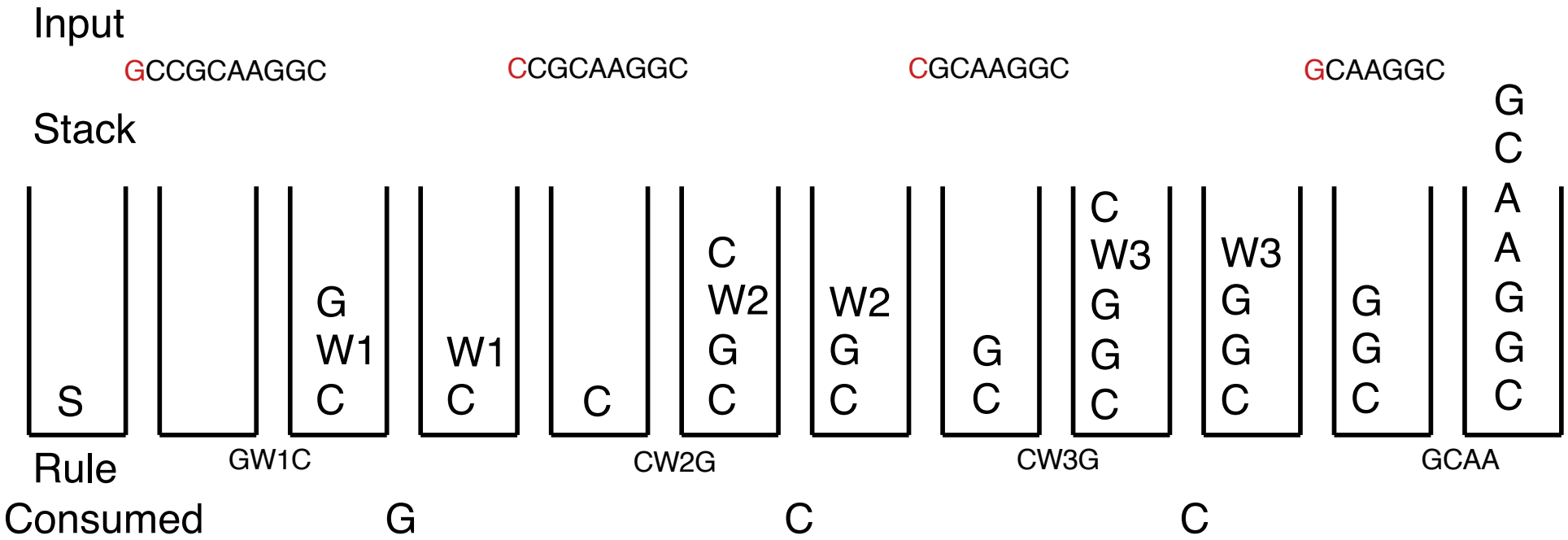
    the following steps are done
    - a next state is chosen
    - a (possibly empty) string of symbols is placed onto the stack, and
    - the input head is advanced by one symbol.
- In the second type of moves, the input head is not advanced at the end of the move as in the case before.

Pushdown automata

Let's take as an example the following CFG that describes an RNA stem-loop with 3 base pairs and GCAA or CCAA in the loop:

S -> AW_1U I CW_1G I GW_1C I UW_1A
W_1 -> AW_2U I CW_2G I GW_2C I UW_2A
W_2 -> AW_3U I CW_3G I GW_3C I UW_3A
W_3 -> GCAA I CCAA

Parsing an RNA stem-loop with a pushdown automaton



… pop the rest of the symbols off the stack and consume the rest of the string.

Pushdown automata

Note than when multiple productions are possible for a symbol the PDA has to evaluate them all individually, (and backtrack).

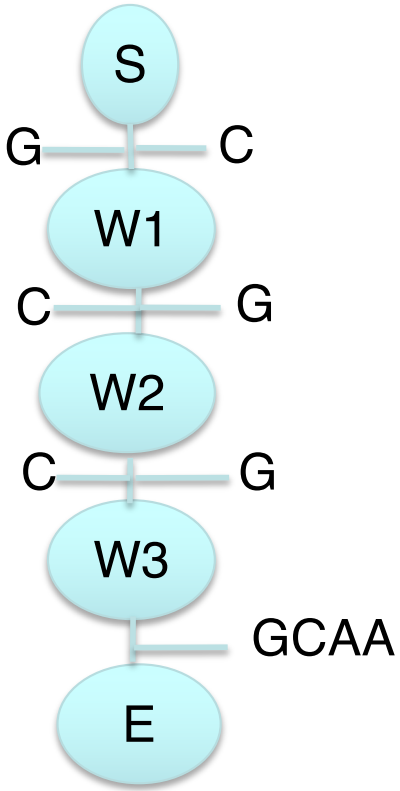
We can use a very similar strategy to generate strings from the grammar.

Basically we have to change the steps so as to randomly choose one of the possible productions for the non-terminal that we pop from the stack.

Parse trees

S -> AW_1U I CW_1G I GW_1C I UW_1A
W_1 -> AW_2U I CW_2G I GW_2C I UW_2A
W_2 -> AW_3U I CW_3G I GW_3C I UW_3A
W_3 -> GCAA I CCAA

The hairpin GCCGCAAGGC has the following parse tree:



A parse tree of a sentence in a given language is the sequence of applications of grammar rules that yield the sentence from the start symbol.

Stochastic grammars

The questions about RNA structures that we would like to address with stochastic context-free grammars are similar to the questions about DNA/RNA sequences that we address with Hidden Markov Models:

(1) Given a set of examples of sequences-structures, estimate optimal parameters for an unparametrized stochastic grammar that represents these sequences-structures (training problem).
Solved with an expectation maximization technique coupled with the equivalent of the forward-backward HMM algorithms for SCFGs, which are called the inside-outside algorithms.

(2) Compute the probability of a sequence given a parametrized stochastic grammar (scoring problem).
Solved by the inside algorithm.

(3) Compute an optimal alignment of a sequence to a parametrized stochastic grammar (alignment problem).
Solved by the Cocke-Younger-Kasami (CYK) algorithm, a variant of the inside algorithm with maximum operations in place of the sums.

Inside/outside algorithms

Assume that we have a sequence x of length L

Further assume that we have a grammar in Chomsky's normal form, meaning that it only has production rules of the form $v \to yz$ and $v \to a$

Emission of a symbol a from state v is denoted by $e_v(a)$

Then the probability of the sequence given the grammar is $\alpha(1, L, S)$ (for convenience we will number the states corresponding to non-terminals as well, so what we need to calculate is $\alpha(1, L, 1)$.

We will do that with an approach we already know from HMMs, using the probabilities $\alpha(i, j, v)$ of parses of subsequence $x[i \ldots j]$ starting in state v, for all i, j, v

Initialization

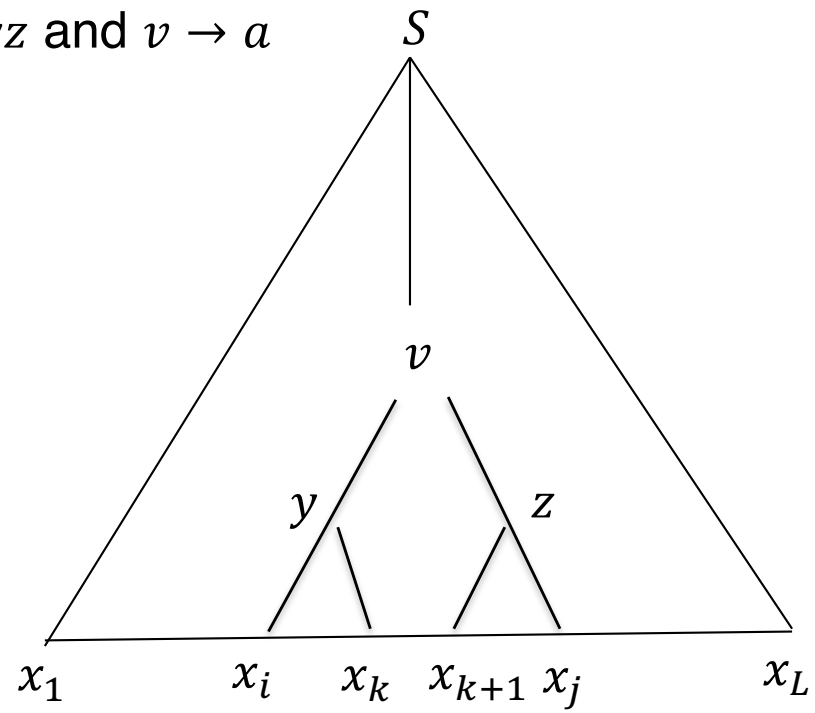
$$
\begin{aligned}
&for(i = 1; i \leq L; i++)\,\{\\
&\quad for(v = 1; v \leq M; v++)\,\{\\
&\qquad\qquad \alpha(i, i, v) = e_v(x_i)\\
&\quad \}\\
&\}
\end{aligned}
$$

Inside algorithm

Remembering that our productions are $v \to yz$ and $v \to a$

Iteration

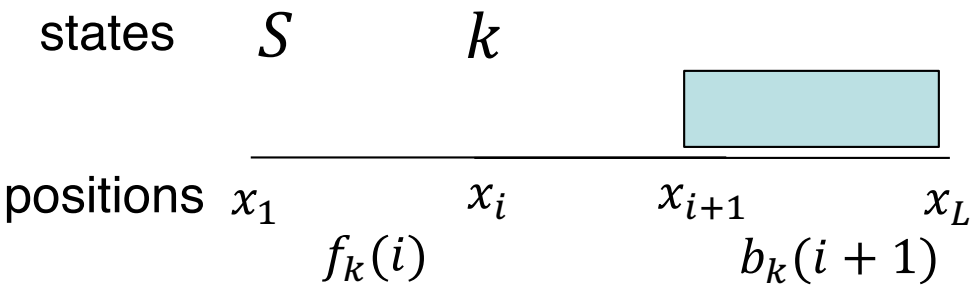
$$
\begin{aligned}
&for(i = 1; i \leq L; i++)\,\{\\
&\quad for(j = i+1; j \leq L; j++)\,\{\\
&\quad\quad for(v = 1; v \leq M; v++)\,\{\\
&\quad\quad\quad \alpha(i,j,v) = 0\\
&\quad\quad\quad for(y = 1; y \leq M; y++)\,\{\\
&\quad\quad\quad\quad for(z = 1; z \leq M; z++)\,\{\\
&\quad\quad\quad\quad\quad for(k = i; k \leq j-1; k++)\,\{\\
&\quad\quad\quad\quad\quad\quad \alpha(i,i,v) \mathrel{+}= \alpha(i,k,y)\alpha(k+1,j,z)\tau_v(yz)\\
&\quad\quad\quad\quad\quad \}\\
&\quad\quad\quad\quad \}\\
&\quad\quad\quad \}\\
&\quad\quad \}\\
&\quad \}\\
&\}
\end{aligned}
$$



Forward/backward vs inside/outside

SRG: Forward
Backward

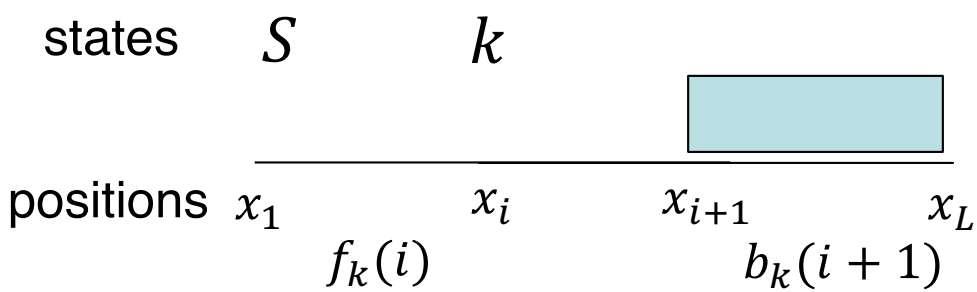
$$P(\pi_i = k|x) = \frac{f_k(i)b_k(i)}{P(x)}$$

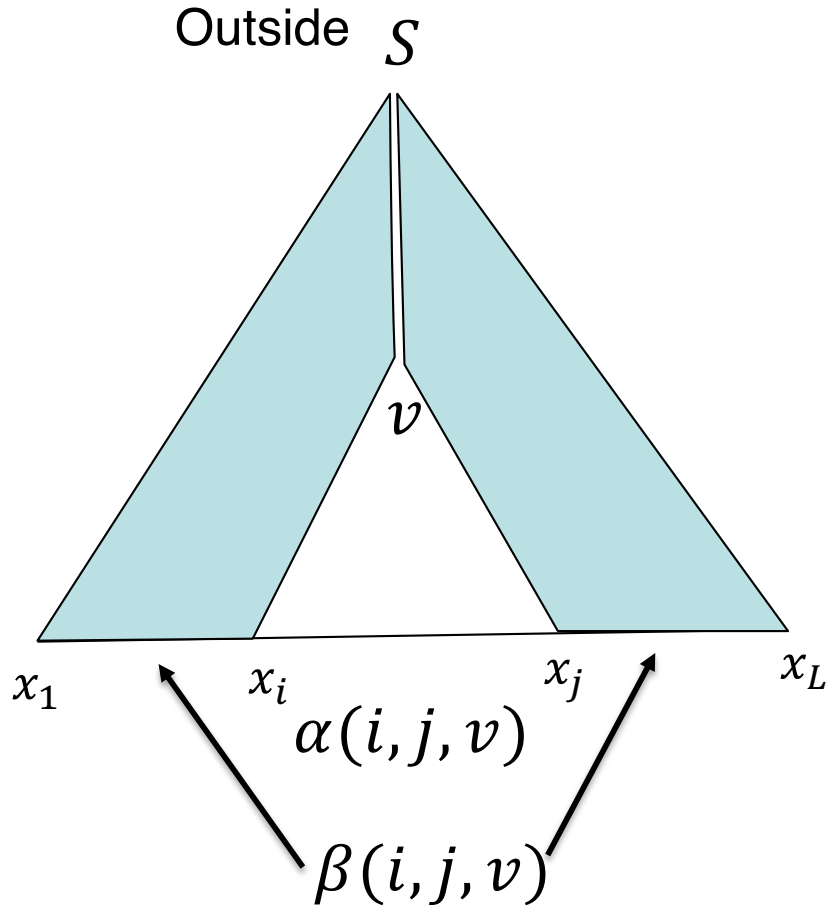


Forward/backward vs inside/outside

SRG: Forward
Backward

SCFG: Inside
Outside





$$P(\pi_i = v|x) = \frac{\alpha(i,i,v)\beta(i,i,v)}{P(x)}$$

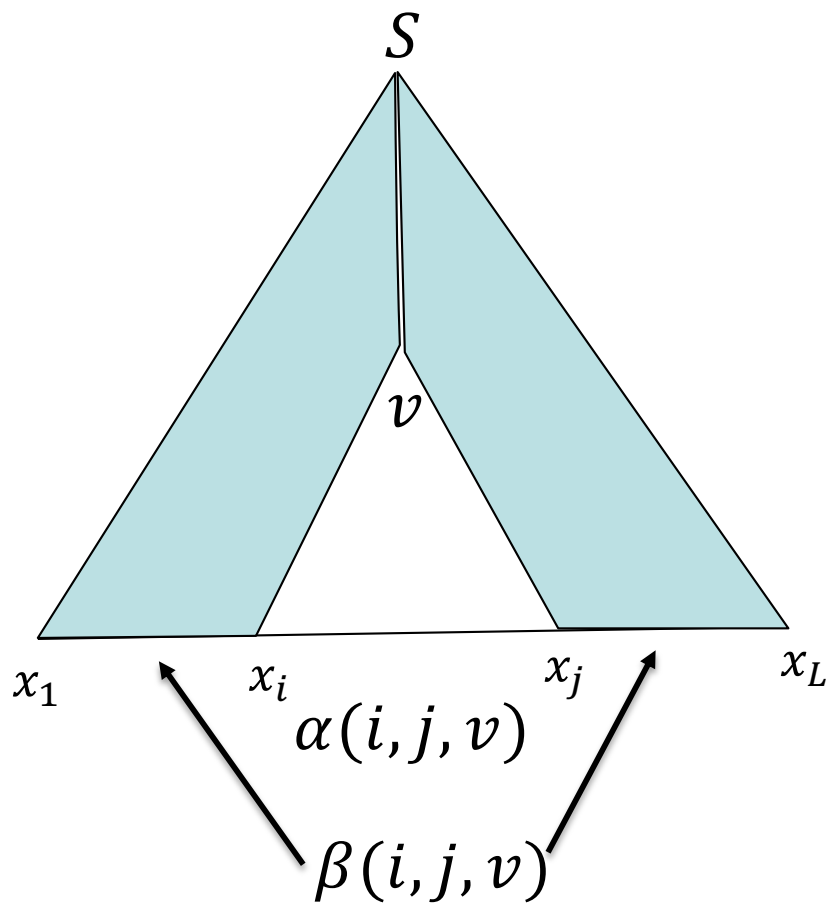
Outside algorithm

The outside algorithm computes the probability $\beta(i, j, v)$ of a complete parse tree rooted at state S for the complete sequence x, except for the subsequence $x_i \ldots x_j$ and ending in state v.

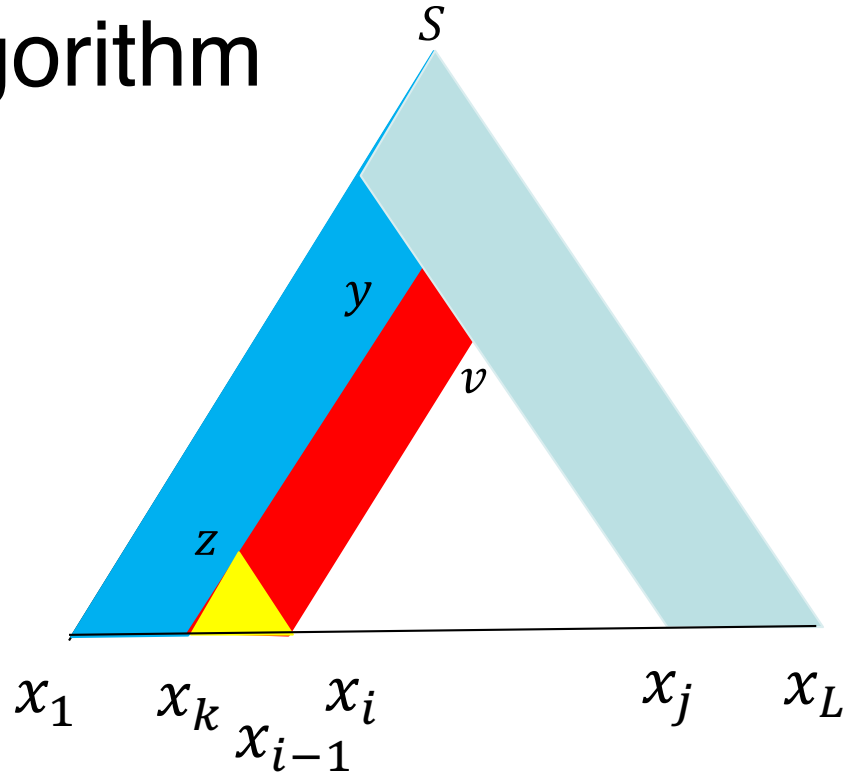
Initialization

$\beta(1, L, 1) = 1$

$\beta(1, L, v) = 0, \forall v = 2 \ldots M$



Outside algorithm



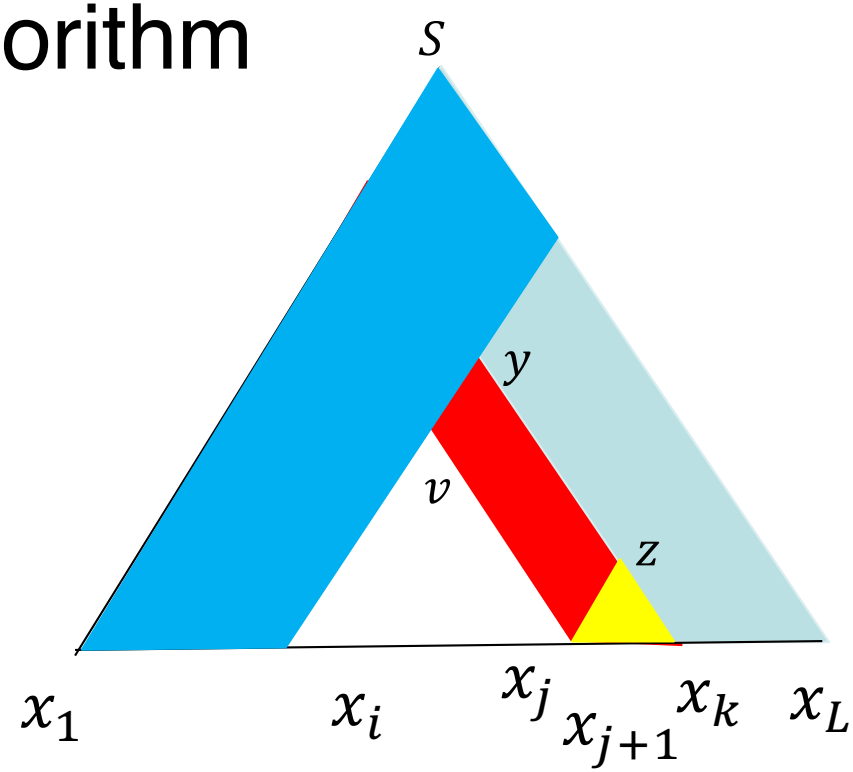
Iteration

$$
\begin{aligned}
&for(i = 1; i \leq L; i++)\{\\
&\quad for(j = L; j \geq i; j--)\{\\
&\quad\quad for(v = 1; v \leq M; v++)\{\\
&\quad\quad\quad \beta(i,j,v) = 0\\
&\quad\quad\quad for(y = 1; y \leq M; y++)\{\\
&\quad\quad\quad\quad for(z = 1; z \leq M; z++)\{\\
&\quad\quad\quad\quad\quad for(k = 1; k \leq i-1; k++)\{\\
&\quad\quad\quad\quad\quad\quad \beta(i,j,v) \mathrel{+}= \alpha(k,i-1,z)\beta(k,j,y)\tau_y(vz)\\
&\quad\quad\quad\quad\quad \}\\
&\quad\quad\quad\quad\quad for(k = j+1; k < L; k++)\{\\
&\quad\quad\quad\quad\quad\quad \beta(i,j,v) \mathrel{+}= \alpha(j+1,k,z)\beta(i,k,y)\tau_y(vz)\\
&\quad\quad\quad\quad\quad \}\\
&\quad\quad\quad\quad \}\\
&\quad\quad\quad \}\\
&\quad\quad \}\\
&\quad \}\\
&\}
\end{aligned}
$$

Outside algorithm

Iteration

$$
\begin{aligned}
&for(i = 1; i \le L; i++)\{\\
&\quad for(j = L; j \ge i; j--)\{\\
&\quad\quad for(v = 1; v \le M; v++)\{\\
&\quad\quad\quad \beta(i,j,v) = 0\\
&\quad\quad\quad for(y = 1; y \le M; y++)\{\\
&\quad\quad\quad\quad for(z = 1; z \le M; z++)\{\\
&\quad\quad\quad\quad\quad for(k = 1; k \le i-1; k++)\{\\
&\quad\quad\quad\quad\quad\quad \beta(i,j,v) \mathrel{+}= \alpha(k, i-1, z)\beta(k,j,y)\tau_y(vz)\\
&\quad\quad\quad\quad\quad \}\\
&\quad\quad\quad\quad\quad for(k = 1; k \le i-1; k++)\{\\
&\quad\quad\quad\quad\quad\quad \beta(i,j,v) \mathrel{+}= \alpha(j+1, k, z)\beta(i,k,y)\tau_y(vz)\\
&\quad\quad\quad\quad\quad \}\\
&\quad\quad\quad\quad \}\\
&\quad\quad\quad \}\\
&\quad\quad \}\\
&\quad \}\\
&\}
\end{aligned}
$$



Training the parameters of a SCFG

In the E-step we determine how many times each non-terminal and each production is used in parses.

In the M-step we update production probabilities.

For a SCFG in Chomsky's normal form $v \to yz$ and $v \to a$

$$\hat{e}_v(a) = \frac{\sum_{\{i|x_i=a\}} \beta(i,i,v) e_v(a)}{\sum_{i=1}^{L} \sum_{j=1}^{L} \beta(i,j,v)\alpha(i,j,v)}$$

$$\hat{t}_v(y,z) = \frac{\sum_{i=1}^{L-1} \sum_{j=i+1}^{L} \sum_{k=i}^{j-1} t_v(y,z)\beta(i,j,v)\alpha(i,k,y)\alpha(k+1,j,z)}{\sum_{i=1}^{L} \sum_{j=1}^{L} \beta(i,j,v)\alpha(i,j,v)}$$