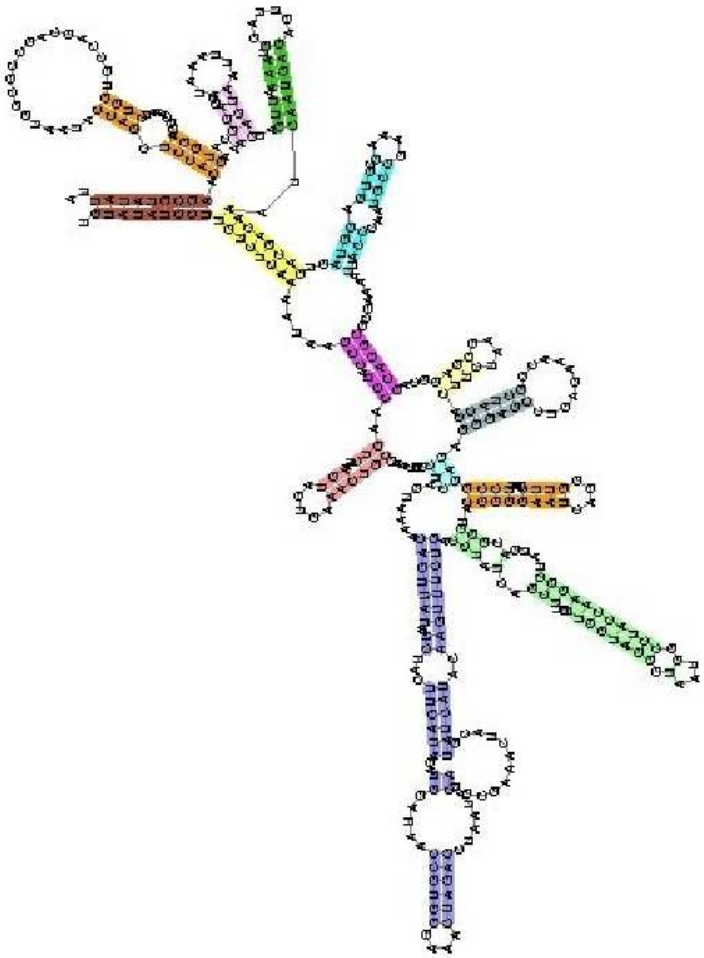
Lecture 5: RNA folding

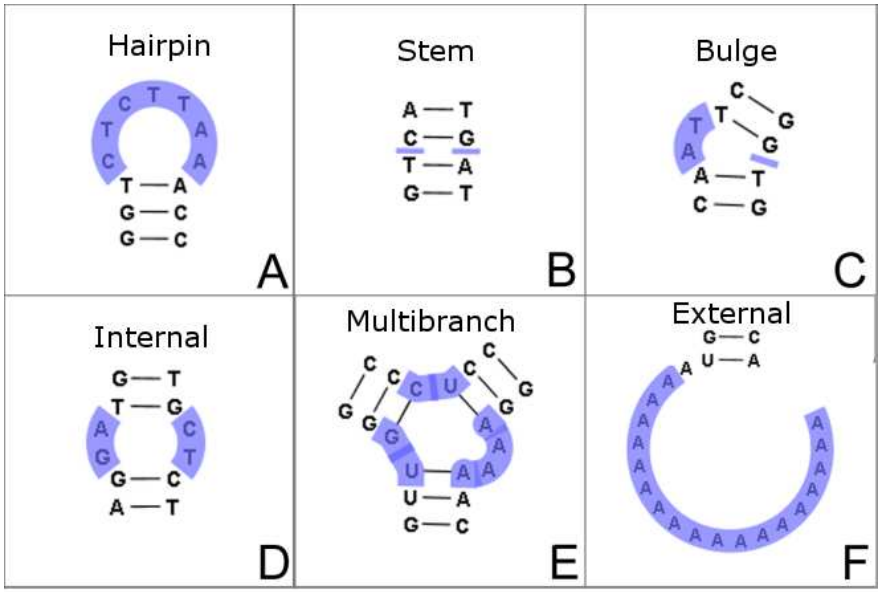
Hamidreza Chitsaz

Colorado State University
chitsaz@cs.colostate.edu

Spring 2020
February 11,13, 2020

Nearest Neighbor Model





McCaskill's Algorithm for MFE Structure (1990)

Notation

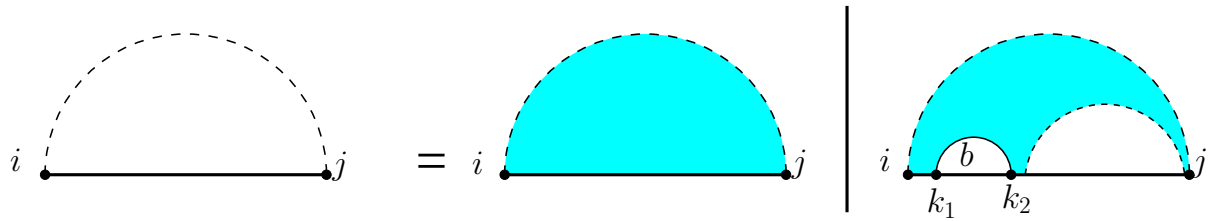
- **straight horizontal line:** nucleotides indexed from 1 to n.
- **solid arc:** a base pair.
- **dashed arc:** can be base pair or not.

- **white region:** open to more recursions.
- **cyan region:** finalized in the recursion, compute its energy contribution.

- **MFE** stands for *minimum free energy*.

McCaskill's Algorithm for MFE Structure (1990)

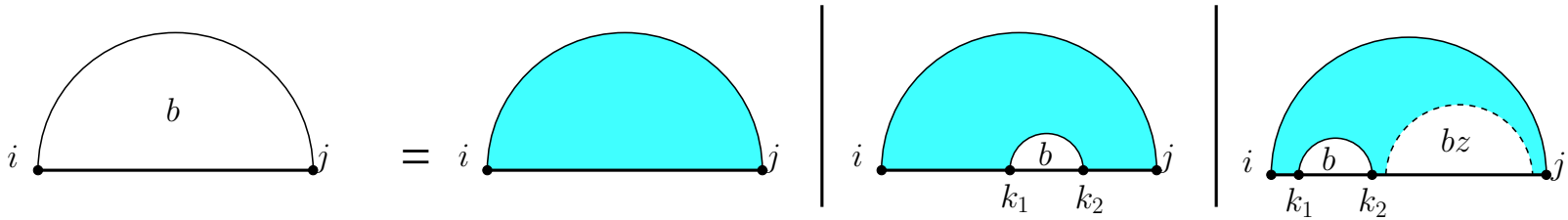
General Case



$$S(i,j) = \min \begin{cases} G_{\text{unfolded}}(i,j) \\ \\ G_{\text{unfolded}}(i,k_1-1) + S_b(k_1,k_2) + S(k_2+1,j) \\ \text{for } i \le k_1 < k_2 \le j \end{cases}$$

McCaskill's Algorithm for MFE Structure (1990)

Stack/Loop Case

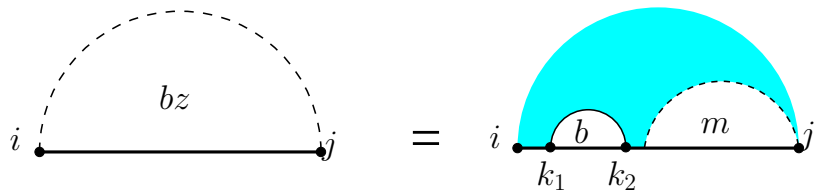


$$S_b(i,j) = \min \begin{cases} G_{\text{hairpin}}(i,j) \\ G_{\text{stack/bulge/int}}(i,k_1,k_2,j) + S_b(k_1,k_2) \\ a_1 + a_2(k_1 - i - 1) + 2a_3 + S_b(k_1,k_2) + S_{bz}(k_2+1, j-1) \\ \text{for } i < k_1 < k_2 < j \end{cases}$$

Recall that multiloop energy is $a_1 + a_2 U + a_3 P$.

McCaskill's Algorithm for MFE Structure (1990)

Multiloop Case

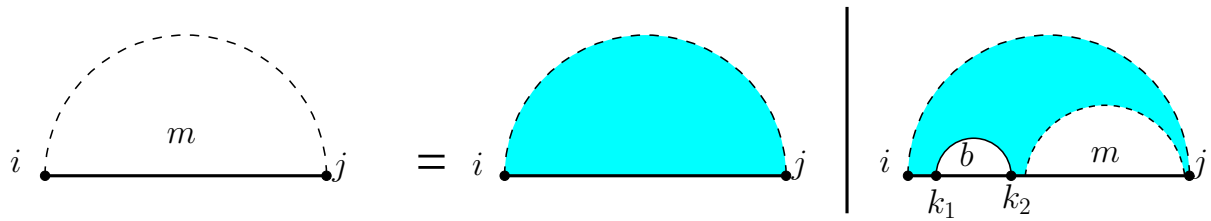


$$S_{bz}(i,j) = \min \begin{cases} a_2(k_1 - i) + a_3 + S_b(k_1, k_2) + S_m(k_2 + 1, j) \\ \text{for } i \leq k_1 < k_2 \leq j \end{cases}$$

Recall that multiloop energy is $a_1 + a_2 U + a_3 P$.

McCaskill's Algorithm for MFE Structure (1990)

Multiloop Case (continued)



$$S_m(i,j) = \min \begin{cases} a_2(j-i+1) \\ a_2(k_1-i) + a_3 + S_b(k_1,k_2) + S_m(k_2+1,j) \\ \text{for } i \le k_1 < k_2 \le j \end{cases}$$

Recall that multiloop energy is $a_1 + a_2 U + a_3 P$.

Ahhh...but MFE is often biologically not-so-relevant!

Question: how about

1. computing base pairing probabilities,
2. sampling from the Boltzmann ensemble structures, clustering, centroids, etc.,
3. and computing equilibrium concentrations and melting temperature?

Answer: the key enabling technology is the **partition function**. All of the above can be computed from the partition function.

Ahhh...but MFE is often biologically not-so-relevant!

Question: how about

1. computing base pairing probabilities,
2. sampling from the Boltzmann ensemble structures, clustering, centroids, etc.,
3. and computing equilibrium concentrations and melting temperature?

Answer: the key enabling technology is the **partition function**. All of the above can be computed from the partition function.

Ahhh...but MFE is often biologically not-so-relevant!

Question: how about

1. computing base pairing probabilities,
2. sampling from the Boltzmann ensemble structures, clustering, centroids, etc.,
3. and computing equilibrium concentrations and melting temperature?

Answer: the key enabling technology is the **partition function**. All of the above can be computed from the partition function.

Ahhh...but MFE is often biologically not-so-relevant!

Question: how about

1. computing base pairing probabilities,
2. sampling from the Boltzmann ensemble structures, clustering, centroids, etc.,
3. and computing equilibrium concentrations and melting temperature?

Answer: the key enabling technology is the **partition function**. All of the above can be computed from the partition function.

Partition Function

$$Q(T) = \sum_{f \in F} e^{-G_f/RT},$$

F : All permissible foldings, i.e. the Boltzmann ensemble,

T : Temperature,

R : Gas constant,

$$p(f) \propto e^{-G_f/RT},$$

and Q is the normalizing factor. Also other thermodynamic quantities can be derived from Q.

Partition Function

$$Q(T) = \sum_{f \in F} e^{-G_f/RT},$$

F : All permissible foldings, i.e. the Boltzmann ensemble,

T : Temperature,

R : Gas constant,

$$p(f) \propto e^{-G_f/RT},$$

and Q is the normalizing factor. Also other thermodynamic quantities can be derived from Q.

Partition Function Hardness $\geq$ MFE Hardness

Partition function

$$\sum_{f \in F} e^{-G_f/RT}.$$

MFE secondary structure

$$\text{argmin}_{f \in F} G_f.$$

Transform any partition function dynamic programming to an MFE algorithm by

$$e^{-G_f} \to G_f$$
$$\times \to +$$
$$\sum \to \min.$$

Partition Function Hardness $\geq$ MFE Hardness

Partition function

$$\sum_{f \in F} e^{-G_f/RT}.$$

MFE secondary structure

$$\text{argmin}_{f \in F} G_f.$$

Transform any partition function dynamic programming to an MFE algorithm by

$$e^{-G_f} \to G_f$$
$$\times \to +$$
$$\sum \to \min.$$

MFE $\rightarrow$ Partition Function

Not always possible

In the partition function

$$\sum_{f \in F} e^{-G_f/RT},$$

every structure f is taken into account *exactly once*, whereas in the structure prediction

$$\text{argmin}_{f \in F} G_f,$$

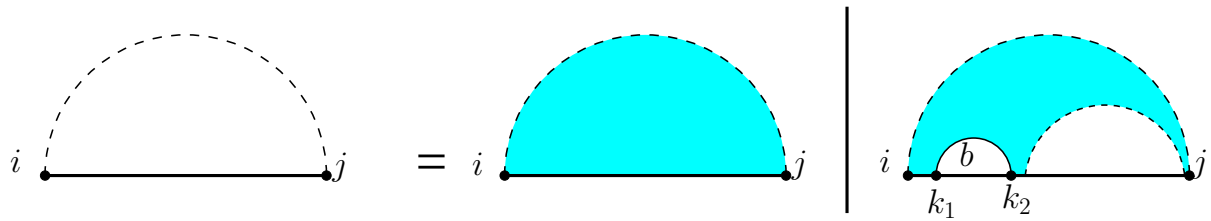
every structure f is taken into account *at least once*.

Transform an *unambiguous* MFE dynamic programming to a partition function algorithm by

$$G_f \rightarrow e^{-G_f}$$
$$+ \rightarrow \times$$
$$\min \rightarrow \sum.$$

McCaskill's Algorithm (1990)

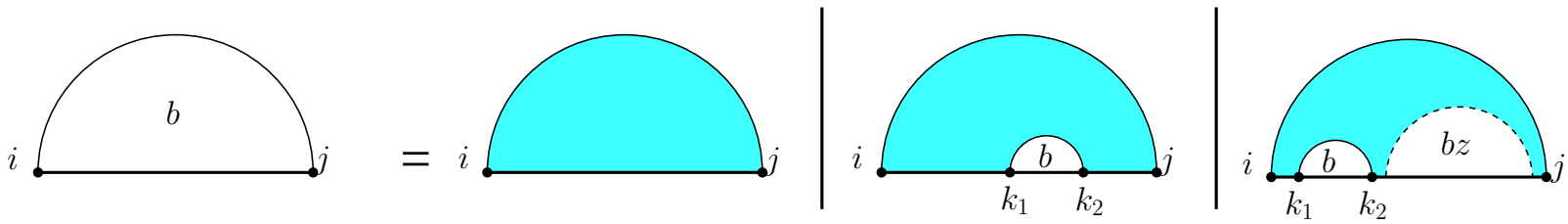
General Case



$$Q(i,j) = e^{-G_{\text{unfolded}}(i,j)/RT} + \sum_{i \le k_1 < k_2 \le j} e^{-G_{\text{unfolded}}(i,k_1-1)/RT} Q_b(k_1,k_2) Q(k_2+1,j).$$

McCaskill's Algorithm (1990)

Stack/Loop Case

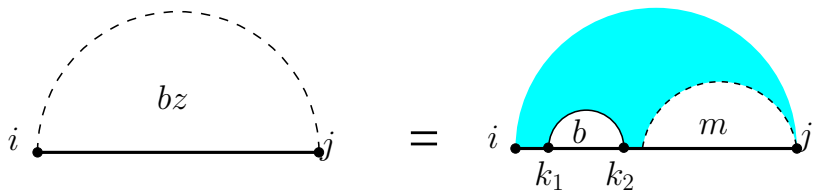


$$Q_b(i,j) = e^{-G_{\text{hairpin}}(i,j)/RT} +$$
$$\sum_{i<k_1<k_2<j} e^{-G_{\text{stack/bulge/int}}(i,k_1,k_2,j)/RT} Q_b(k_1,k_2) +$$
$$\sum_{i<k_1<k_2<j} e^{-[a_1+a_2(k_1-i-1)+2a_3]/RT} Q_b(k_1,k_2) Q_{bz}(k_2+1,j-1).$$

Recall that multiloop energy is $a_1 + a_2 U + a_3 P$.

McCaskill's Algorithm (1990)

Multiloop Case

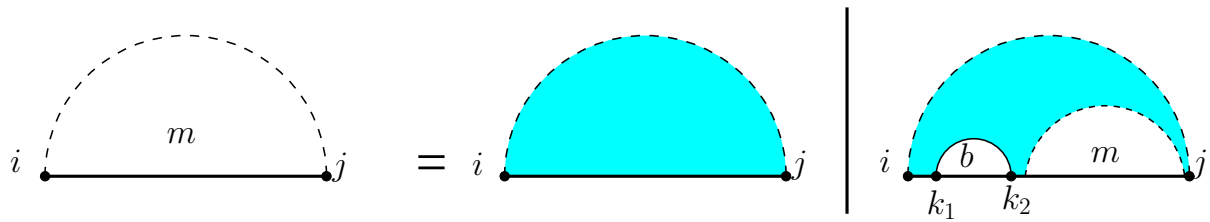


$$Q_{bz}(i,j) = \sum_{i \le k_1 < k_2 \le j} e^{-[a_2(k_1 - i) + a_3]/RT} Q_b(k_1, k_2) Q_m(k_2 + 1, j).$$

Recall that multiloop energy is $a_1 + a_2 U + a_3 P$.

McCaskill's Algorithm (1990)

Multiloop Case (continued)



$$Q_m(i,j) = e^{-a_2(j-i+1)/RT} + \sum_{i \le k_1 < k_2 \le j} e^{-[a_2(k_1-i)+a_3]/RT} Q_b(k_1,k_2) Q_m(k_2+1,j).$$

Recall that multiloop energy is $a_1 + a_2 U + a_3 P$.

Ordering is important

- Pay attention to the order in which $Q(i,j)$, $Q_b(i,j)$, $Q_{bz}(i,j)$, and $Q_m(i,j)$ are computed.
- For instance, Q_m depends on Q_b because it has $Q_b(i,j)$ as a term in its sum.
- Correct ordering: first Q_b then Q_m, Q_{bz}, and Q in parallel.