Algorithmics and combinatorics of RNA sampling

Danièle Gardi[‡] Andy Lorenz[†] Yann Ponty[⋆]

‡ Université Versailles St Quentin – France

† Boston College – Boston – USA

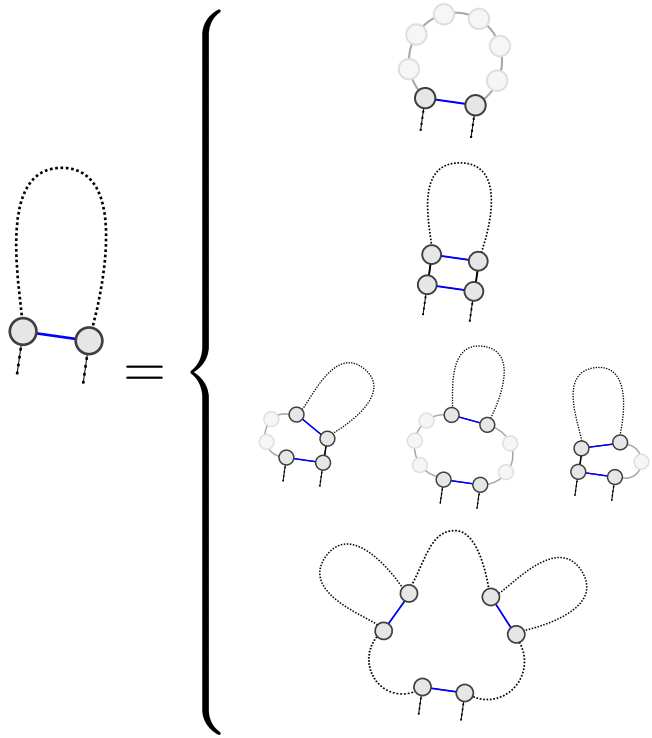
⋆ Polytechnique/CNRS/INRIA AMIB – France

July 29, 2009

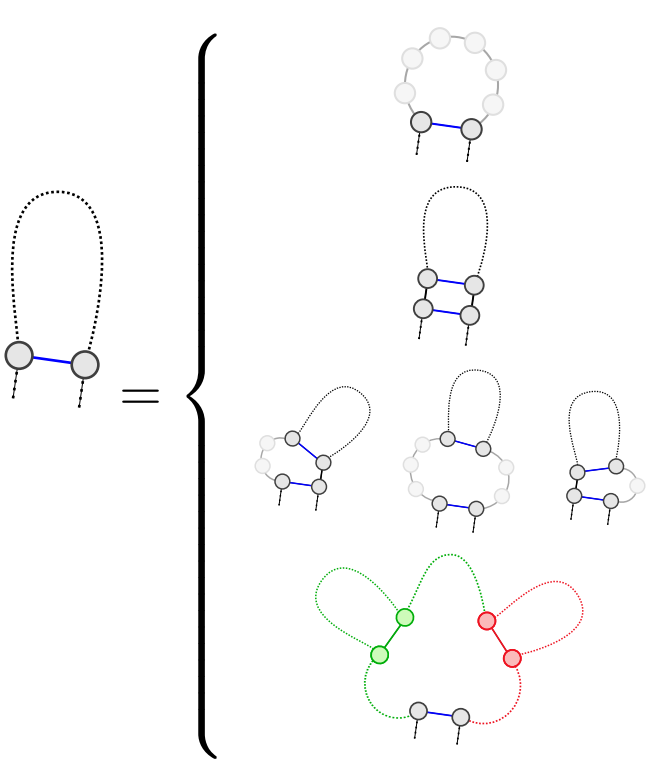
(Enumerative) Combinatorics helps:

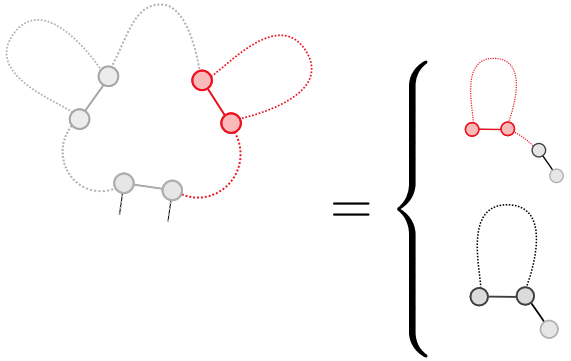
- Counting conformations
- Analyzing features of null models
- Performing analysis of algorithms. . .
- . . . improving them?

MFE DP equations

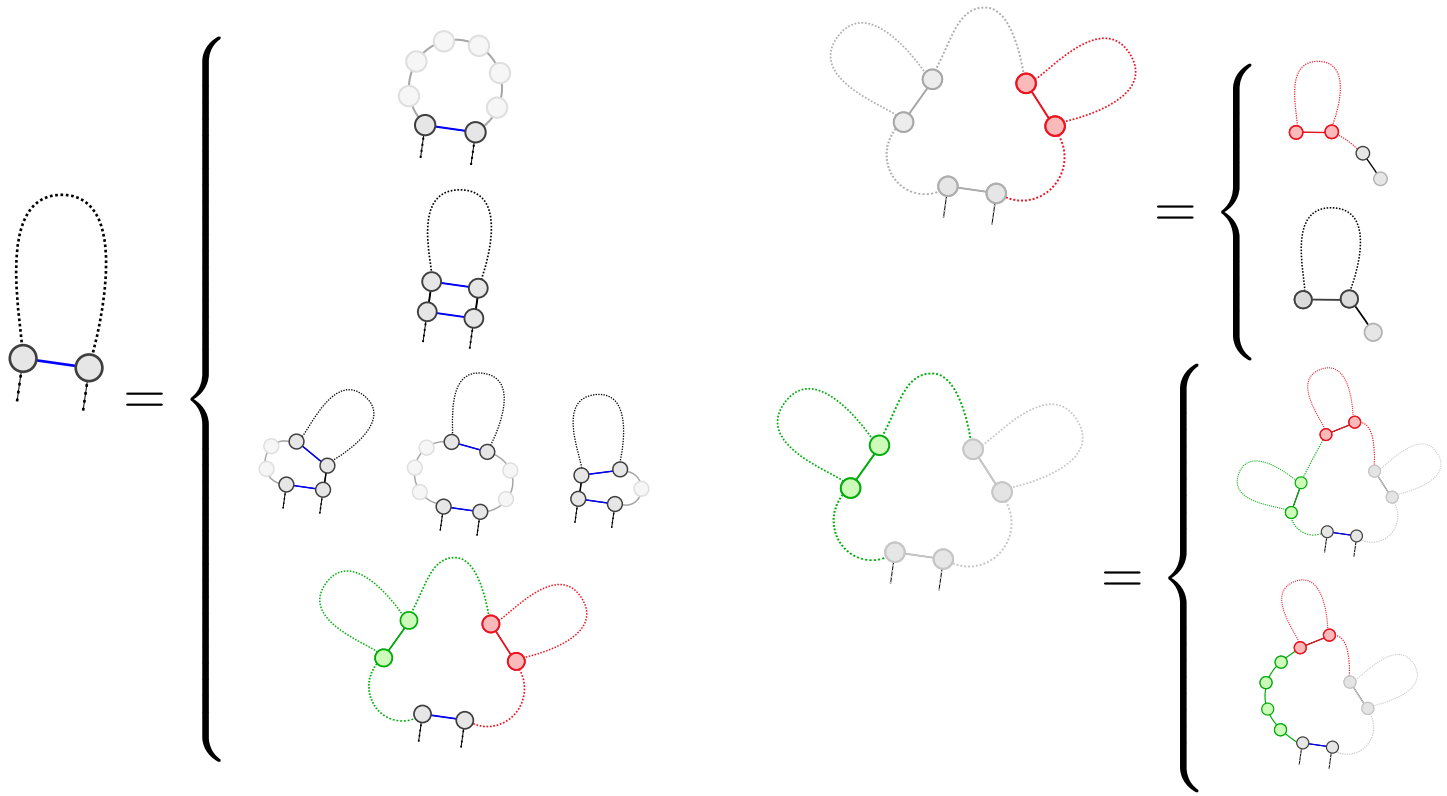


MFE DP equations



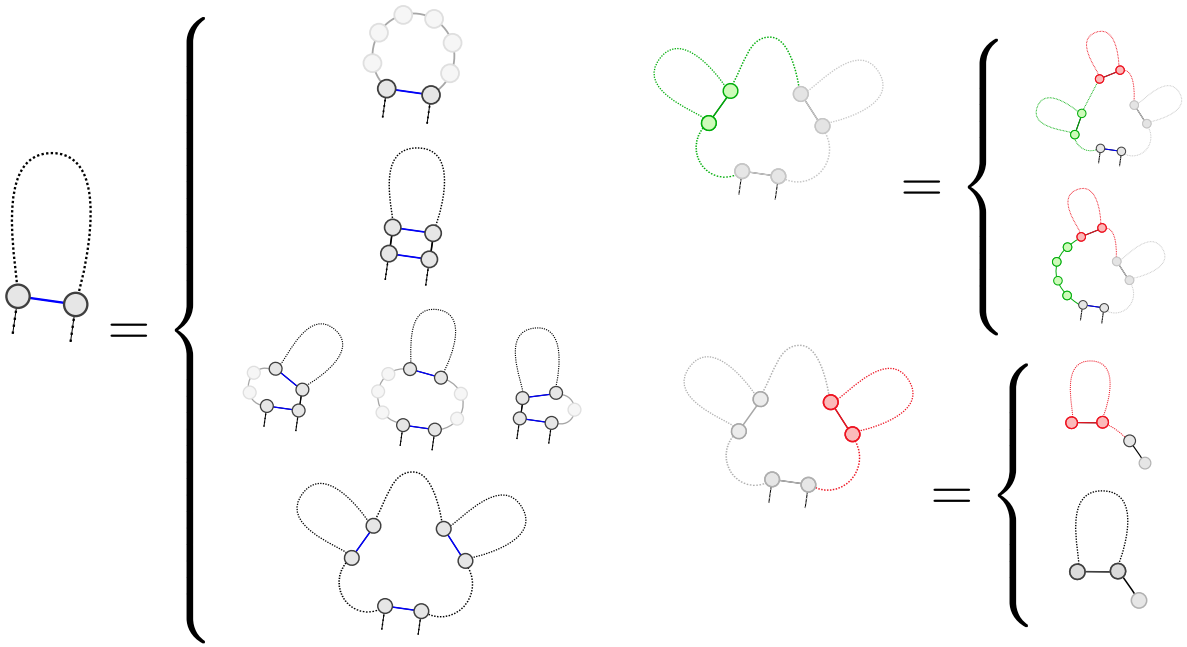


MFE DP equations



Completeness: Not too hard to check!
Unambiguity $\Rightarrow$ Enumerative combinatorics.

Validation



Validation

Generating function $\mathcal{T}(z) = \sum_{n \geq 0} t_n z^n$
With $t_n = \#$Secondary structures of size n

$$\mathcal{A}(z) = \begin{cases} \mathcal{S}(z) \\ z^2\mathcal{A}(z) \\ z\mathcal{S}(z)z^2\mathcal{A}(z) + z^2\mathcal{A}(z)\mathcal{S}(z)z \\ +z\mathcal{S}(z)z^2\mathcal{A}(z)\mathcal{S}(z)z \\ \mathcal{B}(z)\mathcal{C}(z) \end{cases} \qquad \mathcal{B}(z) = \begin{cases} \mathcal{B}(z)\mathcal{C}(z) \\ \mathcal{S}(z)\mathcal{B}(z) \end{cases}$$

$$\mathcal{C}(z) = \begin{cases} \mathcal{C}(z)z \\ z^2\mathcal{A}(z) \end{cases}$$

$$\mathcal{S}(z) = 1 + z\mathcal{S}(z)$$

Validation

$$\mathcal{A}(z) = \begin{cases} \mathcal{S}(z) \\ z^2\mathcal{A}(z) \\ z\mathcal{S}(z)z^2\mathcal{A}(z) + z^2\mathcal{A}(z)\mathcal{S}(z)z \\ +z\mathcal{S}(z)z^2\mathcal{A}(z)\mathcal{S}(z)z \\ \mathcal{B}(z)\mathcal{C}(z) \end{cases} \qquad \mathcal{B}(z) = \begin{cases} \mathcal{B}(z)\mathcal{C}(z) \\ \mathcal{S}(z)\mathcal{B}(z) \end{cases} \qquad \mathcal{C}(z) = \begin{cases} \mathcal{C}(z)z \\ z^2\mathcal{A}(z) \end{cases}$$

$$\mathcal{S}(z) = 1 + z\mathcal{S}(z)$$

Reminder: Waterman counted Sec. Str. [Wat78] and found the gen. fun.

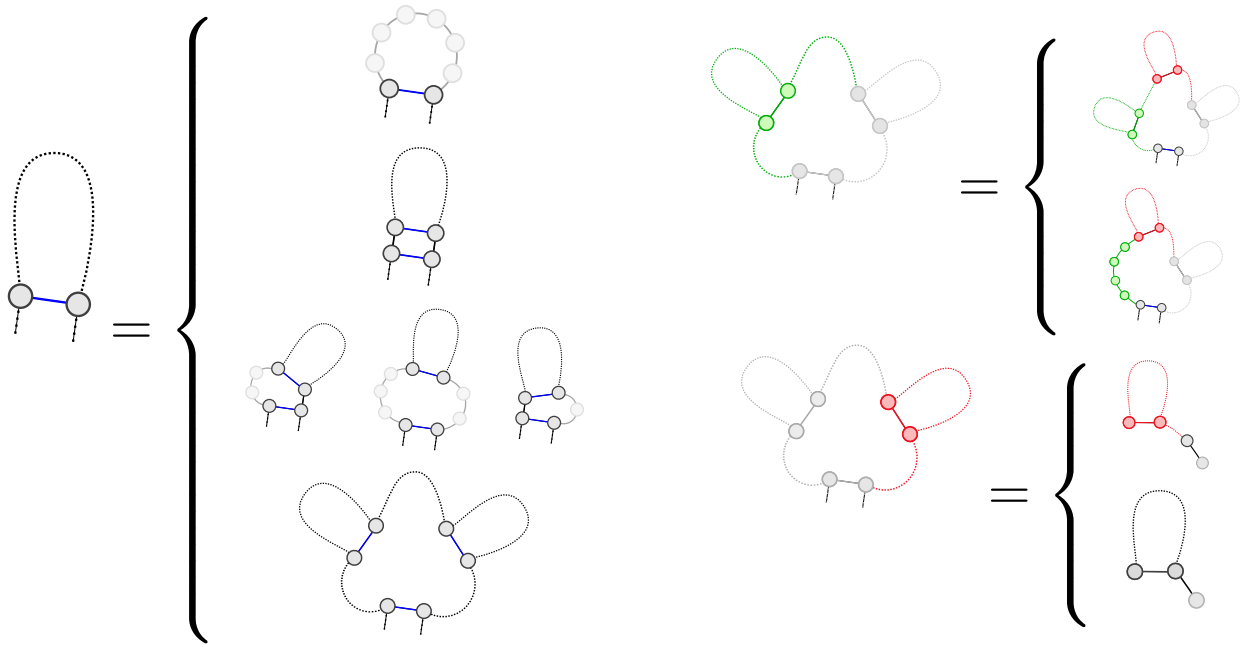
$$\mathcal{W}(z) = \frac{1 - z + z^2 - \sqrt{1 - 2z - z^2 - 2z^3 + z^4}}{2z^2}$$

Here we have

$$\begin{aligned} \Rightarrow \mathcal{A}(z) &= \frac{1 - z - z^2 - \sqrt{1 - 2z - z^2 - 2z^3 + z^4}}{2z^2} \\ &= \mathcal{W}(z) - 1 \quad \text{(Woops, we forgot the } \textit{empty} \text{ RNA)} \end{aligned}$$

MFE folding

- $E_H(i,j)$: Energy of hairpin loop with closing pair (i,j)
- $E_{BI}(i,j)$: Energy of bulge or internal loop with closing pair (i,j)
- $E_S(i,j)$: Energy of stacking pairs $(i,j)/(i+1,j-1)$
- a,c,b: Penalties for multiloop, hairpins and unpaired bases in multiloop.



Message #1

Treating search space as a combinatorial object saves time and trouble!

MFE folding

- $E_H(i,j)$: Energy of hairpin loop with closing pair (i,j)
- $E_{BI}(i,j)$: Energy of bulge or internal loop with closing pair (i,j)
- $E_S(i,j)$: Energy of stacking pairs $(i,j)/(i+1,j-1)$
- a,c,b: Penalties for multiloop, hairpins and unpaired bases in multiloop.

$$
\begin{aligned}
\mathcal{M}'(i,j) &= \text{Min}\left\{\begin{array}{c} E_H(i,j) \\ E_S(i,j)+\mathcal{M}'(i+1,j-1) \\ \text{Min}(E_{BI}(i,i',j',j)+\mathcal{M}'(i',j')) \\ a+c+\text{Min}(\mathcal{M}'(i+1,k-1)+\mathcal{M}^1(k,j-1)) \end{array}\right\} \\
\mathcal{M}(i,j) &= \text{Min}\left\{\text{Min}\left(\mathcal{M}(i,k-1), b(k-1)\right)+\mathcal{M}^1(k,j)\right\} \\
\mathcal{M}^1(i,j) &= \text{Min}\left\{b+\mathcal{M}^1(i,j-1), c+\mathcal{M}'(i,j)\right\}
\end{aligned}
$$

Message #1

Treating search space as a combinatorial object saves time and trouble!

Partition function: Beyond m.f.e. hypothesis

Partition function/Boltzmann probability

- Let ω be an RNA sequence
- $\mathcal{S}_\omega$ be the set of sequences compatible with ω,

$$\textbf{Partition function} \quad \mathcal{Z}_\omega = \sum_{S \in \mathcal{S}_\omega} e^{\frac{-E_{S,\omega}}{RT}}$$

where T is temperature in Kelvin and R is the universal gas constant.

$$\textbf{Boltzmann probability} \quad P_{S,\omega} = \frac{e^{\frac{-E_{S,\omega}}{RT}}}{\mathcal{Z}_\omega}$$

$P_{S,\omega}$ is the probability of observing ω in conformation S.

$\Rightarrow$ Offers a more dynamic view of the folding process

$\Rightarrow$ Gives a model for computing various probabilities (BP, Motifs ...)

$\Rightarrow$ Unified algorithmic framework for subopts and mfe ($RT \to \infty$)

$\Rightarrow$ Very easy to embed into any existing DP equations

Partition function: Beyond m.f.e. hypothesis

Partition function/Boltzmann probability

- Let ω be an RNA sequence
- $\mathcal{S}_\omega$ be the set of sequences compatible with ω,

$$\textbf{Partition function} \quad \mathcal{Z}_\omega = \sum_{S \in \mathcal{S}_\omega} e^{\frac{-E_{S,\omega}}{RT}}$$

where T is temperature in Kelvin and R is the universal gas constant.

$$\textbf{Boltzmann probability} \quad P_{S,\omega} = \frac{e^{\frac{-E_{S,\omega}}{RT}}}{\mathcal{Z}_\omega}$$

$P_{S,\omega}$ is the probability of observing ω in conformation S.

$\Rightarrow$ Offers a more dynamic view of the folding process

$\Rightarrow$ Gives a model for computing various probabilities (BP, Motifs ...)

$\Rightarrow$ Unified algorithmic framework for subopts and mfe ($RT \to \infty$)

$\Rightarrow$ Very easy to embed into any existing DP equations

Partition function: Beyond m.f.e. hypothesis

Partition function/Boltzmann probability

- Let ω be an RNA sequence
- $\mathcal{S}_\omega$ be the set of sequences compatible with ω,

$$\textbf{Partition function} \quad \mathcal{Z}_\omega = \sum_{S \in \mathcal{S}_\omega} e^{\frac{-E_{S,\omega}}{RT}}$$

where T is temperature in Kelvin and R is the universal gas constant.

$$\textbf{Boltzmann probability} \quad P_{S,\omega} = \frac{e^{\frac{-E_{S,\omega}}{RT}}}{\mathcal{Z}_\omega}$$

$P_{S,\omega}$ is the probability of observing ω in conformation S.

$\Rightarrow$ Offers a more dynamic view of the folding process

$\Rightarrow$ Gives a model for computing various probabilities (BP, Motifs ...)

$\Rightarrow$ Unified algorithmic framework for subopts and mfe ($RT \to \infty$)

$\Rightarrow$ Very easy to embed into any existing DP equations

Partition function: Beyond m.f.e. hypothesis

Partition function/Boltzmann probability

- Let ω be an RNA sequence
- $\mathcal{S}_\omega$ be the set of sequences compatible with ω,

$$\textbf{Partition function} \quad \mathcal{Z}_\omega = \sum_{S \in \mathcal{S}_\omega} e^{\frac{-E_{S,\omega}}{RT}}$$

where T is temperature in Kelvin and R is the universal gas constant.

$$\textbf{Boltzmann probability} \quad P_{S,\omega} = \frac{e^{\frac{-E_{S,\omega}}{RT}}}{\mathcal{Z}_\omega}$$

$P_{S,\omega}$ is the probability of observing ω in conformation S.

$\Rightarrow$ Offers a more dynamic view of the folding process

$\Rightarrow$ Gives a model for computing various probabilities (BP, Motifs ...)

$\Rightarrow$ Unified algorithmic framework for subopts and mfe ($RT \to \infty$)

$\Rightarrow$ Very easy to embed into any existing DP equations

Partition function: Beyond m.f.e. hypothesis

Partition function/Boltzmann probability

- Let ω be an RNA sequence
- $\mathcal{S}_\omega$ be the set of sequences compatible with ω,

$$\textbf{Partition function} \quad \mathcal{Z}_\omega = \sum_{S \in \mathcal{S}_\omega} e^{\frac{-E_{S,\omega}}{RT}}$$

where T is temperature in Kelvin and R is the universal gas constant.

$$\textbf{Boltzmann probability} \quad P_{S,\omega} = \frac{e^{\frac{-E_{S,\omega}}{RT}}}{\mathcal{Z}_\omega}$$

$P_{S,\omega}$ is the probability of observing ω in conformation S.

$\Rightarrow$ Offers a more dynamic view of the folding process

$\Rightarrow$ Gives a model for computing various probabilities (BP, Motifs ...)

$\Rightarrow$ Unified algorithmic framework for subopts and mfe ($RT \to \infty$)

$\Rightarrow$ Very easy to embed into any existing DP equations

Partition function: Beyond m.f.e. hypothesis

From m.f.e. folding to partition function [McC90]:

- Atomic energy increment $E \rightarrow$ Boltzmann factor $e^{\frac{-E}{RT}}$
- Energies contr. move to the exponent:
  Sums $(+) \rightarrow$ Products $(\times)$
- Summing instead of minimizing: Min $\rightarrow$ Sums $(\sum)$

$$
\begin{aligned}
\mathcal{M}'(i,j) &= \text{Min}\left\{\begin{array}{c} E_H(i,j) \\ E_S(i,j)+\mathcal{M}'(i+1,j-1) \\ \text{Min}(E_{BI}(i,i',j',j)+\mathcal{M}'(i',j')) \\ a+c+\text{Min}\left(\mathcal{M}'(i+1,k-1)+\mathcal{M}^1(k,j-1)\right) \end{array}\right\} \\
\mathcal{M}(i,j) &= \text{Min}\left\{\text{Min}\left(\mathcal{M}(i,k-1),b(k-1)\right)+\mathcal{M}^1(k,j)\right\} \\
\mathcal{M}^1(i,j) &= \text{Min}\left\{b+\mathcal{M}^1(i,j-1),c+\mathcal{M}'(i,j)\right\}
\end{aligned}
$$

Message #2

From **unambiguous description** partition function (and then statistical sampling) is just one algebra switch $(\text{Min},+) \rightarrow (+,\times)$ away.

Partition function: Beyond m.f.e. hypothesis

From m.f.e. folding to partition function [McC90]:

- Atomic energy increment $E \longrightarrow$ Boltzmann factor $e^{\frac{-E}{RT}}$
- Energies contr. move to the exponent:
  Sums $(+) \longrightarrow$ Products $(\times)$
- Summing instead of minimizing: Min $\longrightarrow$ Sums $(\sum)$

$$
\begin{aligned}
\mathcal{M}'(i,j) &= \text{Min}\left\{
\begin{array}{c}
e^{\frac{-E_H(i,j)}{RT}} \\
e^{\frac{-E_S(i,j)}{RT}} + \mathcal{M}'(i+1,j-1) \\
\text{Min}\left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} + \mathcal{M}'(i',j')\right) \\
e^{\frac{-(a+c)}{RT}} + \text{Min}\left(\mathcal{M}'(i+1,k-1) + \mathcal{M}^1(k,j-1)\right)
\end{array}
\right\} \\
\mathcal{M}(i,j) &= \text{Min}\left\{\text{Min}\left(\mathcal{M}(i,k-1), e^{\frac{-b(k-1)}{RT}}\right) + \mathcal{M}^1(k,j)\right\} \\
\mathcal{M}^1(i,j) &= \text{Min}\left\{e^{\frac{-b}{RT}} + \mathcal{M}^1(i,j-1), e^{\frac{-c}{RT}} + \mathcal{M}'(i,j)\right\}
\end{aligned}
$$

Message #2

From **unambiguous description** partition function (and then statistical sampling) is just one algebra switch $(\text{Min}, +) \longrightarrow (+, \times)$ away.

Partition function: Beyond m.f.e. hypothesis

From m.f.e. folding to partition function [McC90]:

- Atomic energy increment $E \longrightarrow$ Boltzmann factor $e^{\frac{-E}{RT}}$
- Energies contr. move to the exponent:
  Sums $(+) \longrightarrow$ Products $(\times)$
- Summing instead of minimizing: Min $\longrightarrow$ Sums $(\sum)$

$$\mathcal{M}'(i,j) = \text{Min}\left\{\begin{array}{c} e^{\frac{-E_H(i,j)}{RT}} \\ e^{\frac{-E_S(i,j)}{RT}}\mathcal{M}'(i+1,j-1) \\ \text{Min}\left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}}\mathcal{M}'(i',j')\right) \\ e^{\frac{-(a+c)}{RT}}\text{Min}\left(\mathcal{M}'(i+1,k-1)\mathcal{M}^1(k,j-1)\right) \end{array}\right\}$$

$$\mathcal{M}(i,j) = \text{Min}\left\{\text{Min}\left(\mathcal{M}(i,k-1), e^{\frac{-b(k-1)}{RT}}\right)\mathcal{M}^1(k,j)\right\}$$

$$\mathcal{M}^1(i,j) = \text{Min}\left\{e^{\frac{-b}{RT}}\mathcal{M}^1(i,j-1), e^{\frac{-c}{RT}}\mathcal{M}'(i,j)\right\}$$

Message #2

From **unambiguous description** partition function (and then statistical sampling) is just one algebra switch $(\text{Min}, +) \longrightarrow (+, \times)$ away.

Partition function: Beyond m.f.e. hypothesis

From m.f.e. folding to partition function [McC90]:

- Atomic energy increment $E \longrightarrow$ Boltzmann factor $e^{\frac{-E}{RT}}$
- Energies contr. move to the exponent:
  Sums $(+) \longrightarrow$ Products $(\times)$
- Summing instead of minimizing: Min $\longrightarrow$ Sums $(\sum)$

$$
\begin{aligned}
\mathcal{Z}'(i,j) &= \sum \left\{ \begin{array}{c} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) \\ + \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) \\ + e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1)\mathcal{Z}^1(k,j-1) \right) \end{array} \right\} \\
\mathcal{Z}(i,j) &= \sum \left(\mathcal{Z}(i,k-1) + e^{\frac{-b(k-1)}{RT}} \right) \mathcal{Z}^1(k,j) \\
\mathcal{Z}^1(i,j) &= e^{\frac{-b}{RT}} \mathcal{Z}^1(i,j-1) + e^{\frac{-c}{RT}} \mathcal{Z}'(i,j)
\end{aligned}
$$

Message #2

From **unambiguous description** partition function (and then statistical sampling) is just one algebra switch (Min, $+$) $\longrightarrow$ ($+$, $\times$) away.

Partition function: Beyond m.f.e. hypothesis

From m.f.e. folding to partition function [McC90]:

- Atomic energy increment $E \longrightarrow$ Boltzmann factor $e^{\frac{-E}{RT}}$
- Energies contr. move to the exponent:
  Sums $(+) \longrightarrow$ Products $(\times)$
- Summing instead of minimizing: Min $\longrightarrow$ Sums $(\sum)$

$$
\begin{aligned}
\mathcal{Z}'(i,j) &= \sum \left\{ \begin{array}{c} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) \\ + \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) \\ + e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) \end{array} \right\} \\
\mathcal{Z}(i,j) &= \sum \left(\mathcal{Z}(i,k-1) + e^{\frac{-b(k-1)}{RT}} \right) \mathcal{Z}^1(k,j) \\
\mathcal{Z}^1(i,j) &= e^{\frac{-b}{RT}} \mathcal{Z}^1(i,j-1) + e^{\frac{-c}{RT}} \mathcal{Z}'(i,j)
\end{aligned}
$$

Message #2

From **unambiguous description** partition function (and then statistical sampling) is just one algebra switch $(\text{Min}, +) \longrightarrow (+, \times)$ away.

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

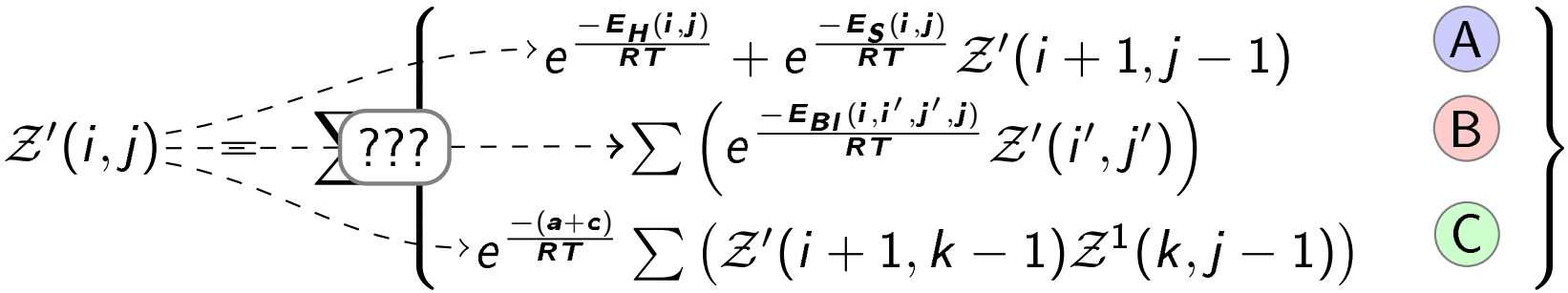
1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

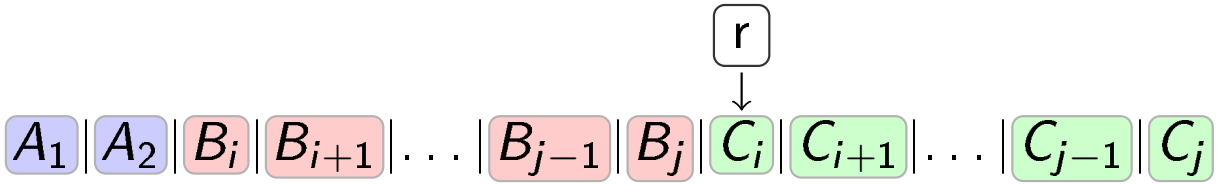


Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$



After $\Theta(n)$ operations, recurse over size $n-1$ interval
$\Rightarrow$ Worst-case time complexity for k samples in $\mathcal{O}(n^2 k)$

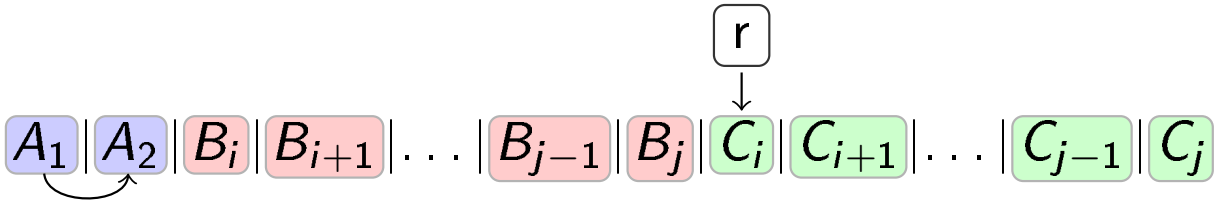
Remark: This is a weighted instance of the so-called recursive random generation of decomposable objects [DRT00].

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$



After $\Theta(n)$ operations, recurse over size $n-1$ interval
$\Rightarrow$ Worst-case time complexity for k samples in $\mathcal{O}(n^2 k)$

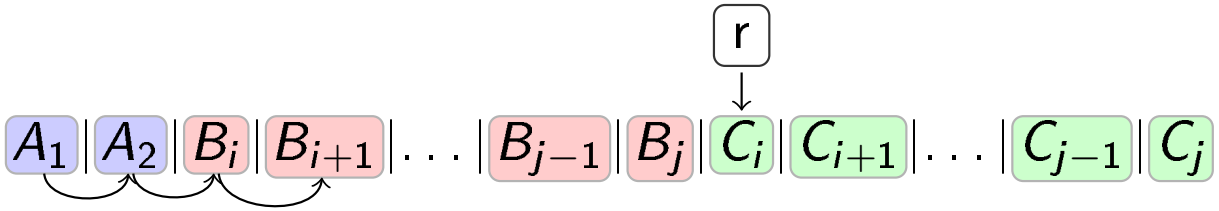
Remark: This is a weighted instance of the so-called recursive random generation of decomposable objects [DRT00].

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$



After $\Theta(n)$ operations, recurse over size $n-1$ interval
$\Rightarrow$ Worst-case time complexity for k samples in $\mathcal{O}(n^2 k)$

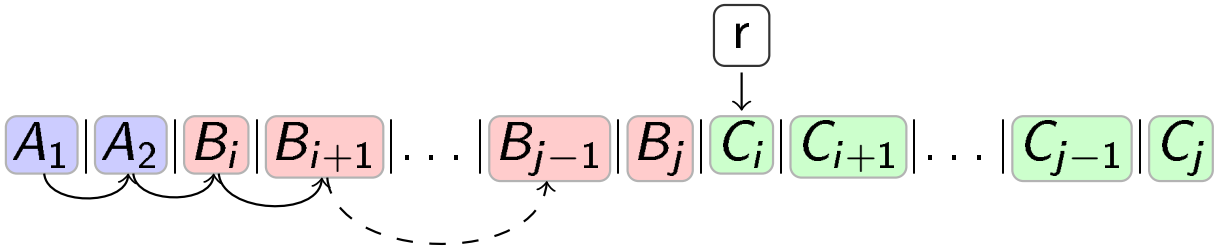
Remark: This is a weighted instance of the so-called recursive random generation of decomposable objects [DRT00].

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$



After $\Theta(n)$ operations, recurse over size $n-1$ interval
$\Rightarrow$ Worst-case time complexity for k samples in $\mathcal{O}(n^2 k)$

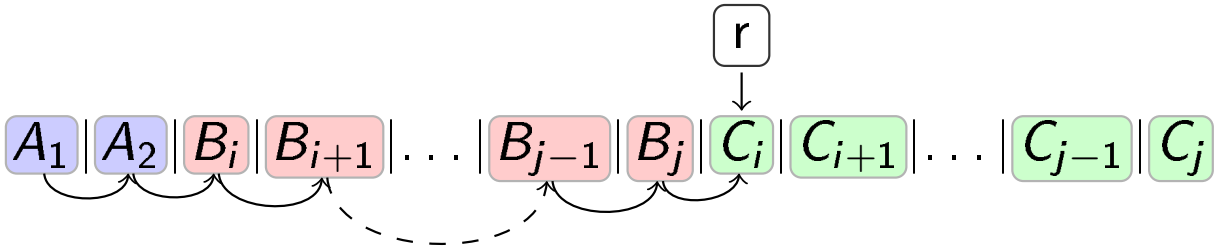
Remark: This is a weighted instance of the so-called recursive random generation of decomposable objects [DRT00].

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$



After $\Theta(n)$ operations, recurse over size $n-1$ interval
$\Rightarrow$ Worst-case time complexity for k samples in $\mathcal{O}(n^2 k)$

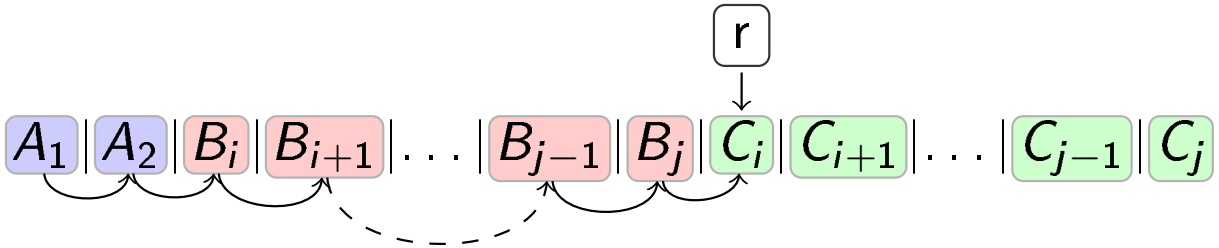
Remark: This is a weighted instance of the so-called recursive random generation of decomposable objects [DRT00].

Statistical sampling through stochastic traceback

Algorithm `SFold` [DL03]:

1. Generate a random number in $[0, \mathcal{Z}'(i,j))$
2. Subtract to r individual contributions to $\mathcal{Z}'(i,j)$, until $r < 0$
3. Recurse over substructures

$$\mathcal{Z}'(i,j) = \sum \left\{ \begin{array}{cc} e^{\frac{-E_H(i,j)}{RT}} + e^{\frac{-E_S(i,j)}{RT}} \mathcal{Z}'(i+1,j-1) & \text{A} \\ \sum \left(e^{\frac{-E_{BI}(i,i',j',j)}{RT}} \mathcal{Z}'(i',j') \right) & \text{B} \\ e^{\frac{-(a+c)}{RT}} \sum \left(\mathcal{Z}'(i+1,k-1) \mathcal{Z}^1(k,j-1) \right) & \text{C} \end{array} \right\}$$



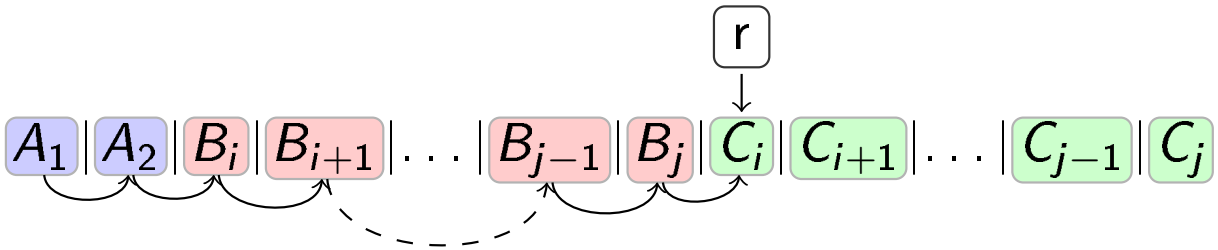
After $\Theta(n)$ operations, recurse over size $n-1$ interval
$\Rightarrow$ Worst-case time complexity for k samples in $\mathcal{O}(n^2 k)$

Remark: This is a weighted instance of the so-called recursive random generation of decomposable objects [DRT00].

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity:
  Average-case time complexity in $\Theta(kn\sqrt{n})$ [Pon08]
  ($\Theta(n^2)$ arises from recursing on $n - \mathcal{O}(1)$ after $\Theta(n)$ ops)
  - Interleaving Bulges (B) and Multiloops (C) contributions
  - Boustrophedon [FZV94]
    Investigate uneven decompositions first, then even ones !
- Non-redundant generation



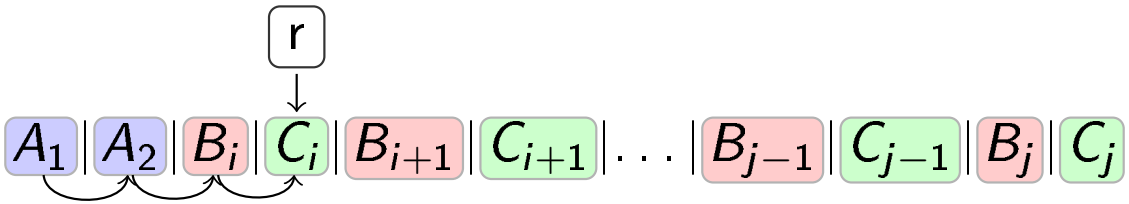
Message #3

Boustrophedon search saves $\Theta(\frac{n}{\log n})/\Omega(\frac{\sqrt{n}}{\log n})$ worst/average case.

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity:
  Average-case time complexity in $\Theta(kn\sqrt{n})$ [Pon08]
  ($\Theta(n^2)$ arises from recursing on $n - \mathcal{O}(1)$ after $\Theta(n)$ ops)
  - Interleaving Bulges (B) and Multiloops (C) contributions
  - Boustrophedon [FZV94]
    Investigate uneven decompositions first, then even ones !
- Non-redundant generation



$\Rightarrow$ Some terms from B and C are reached in $\mathcal{O}(1)$

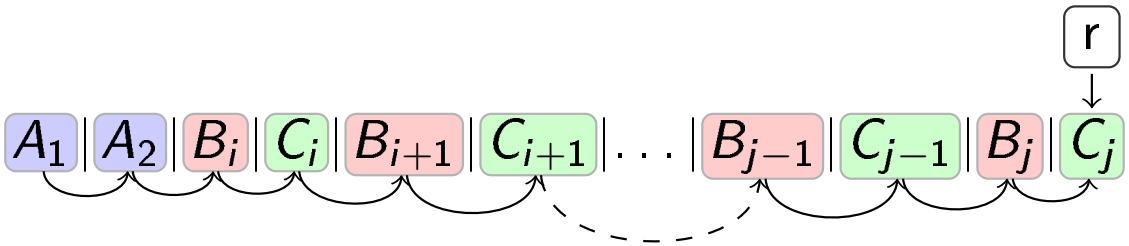
Message #3

Boustrophedon search saves $\Theta(\frac{n}{\log n})/\Omega(\frac{\sqrt{n}}{\log n})$ worst/average case.

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity:
  Average-case time complexity in $\Theta(kn\sqrt{n})$ [Pon08]
  ($\Theta(n^2)$ arises from recursing on $n - \mathcal{O}(1)$ after $\Theta(n)$ ops)
  - Interleaving Bulges (B) and Multiloops (C) contributions
  - Boustrophedon [FZV94]
    Investigate uneven decompositions first, then even ones !
- Non-redundant generation



$\Rightarrow$ Some terms from B and C are reached in $\mathcal{O}(1)$
But still $\Theta(n^2)$, since $\mathcal{Z}'(i,j) \rightarrow (\mathcal{Z}'(i+1,k-1), \mathcal{Z}^1(k,j-1))$

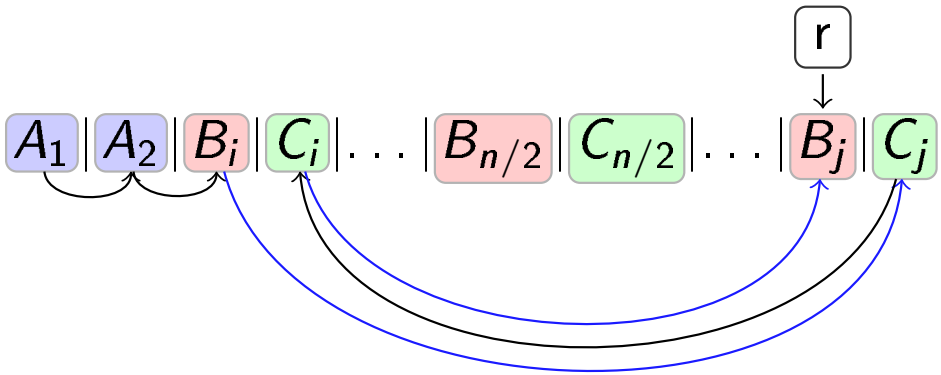
Message #3

Boustrophedon search saves $\Theta(\frac{n}{\log n})/\Omega(\frac{\sqrt{n}}{\log n})$ worst/average case.

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity:
  Average-case time complexity in $\Theta(kn\sqrt{n})$ [Pon08]
  ($\Theta(n^2)$ arises from recursing on $n - \mathcal{O}(1)$ after $\Theta(n)$ ops)
  - Interleaving Bulges (B) and Multiloops (C) contributions
  - Boustrophedon [FZV94]
    Investigate uneven decompositions first, then even ones !
- Non-redundant generation



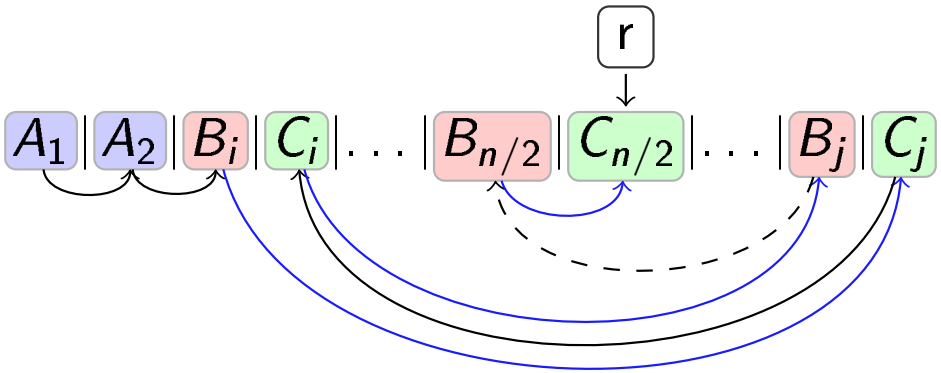
Message #3

Boustrophedon search saves $\Theta(\frac{n}{\log n})/\Omega(\frac{\sqrt{n}}{\log n})$ worst/average case.

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity:
  Average-case time complexity in $\Theta(kn\sqrt{n})$ [Pon08]
  ($\Theta(n^2)$ arises from recursing on $n - \mathcal{O}(1)$ after $\Theta(n)$ ops)
  - Interleaving Bulges (B) and Multiloops (C) contributions
  - Boustrophedon [FZV94] $\Rightarrow \Theta(n \log(n))$ worst-case
    Investigate uneven decompositions first, then even ones !
- Non-redundant generation



Worst-case: Divide evenly at each step [GK81] $\Rightarrow \Theta(n \log(n))$

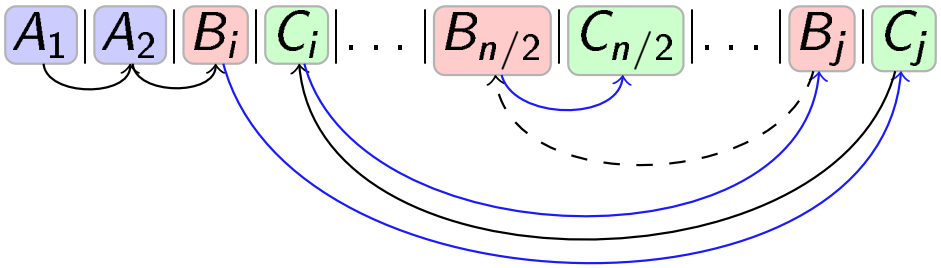
Message #3

Boustrophedon search saves $\Theta(\frac{n}{\log n})/\Omega(\frac{\sqrt{n}}{\log n})$ worst/average case.

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity:
  Average-case time complexity in $\Theta(kn\sqrt{n})$ [Pon08]
  ($\Theta(n^2)$ arises from recursing on $n - \mathcal{O}(1)$ after $\Theta(n)$ ops)
  - Interleaving Bulges (B) and Multiloops (C) contributions
  - Boustrophedon [FZV94]
    Investigate uneven decompositions first, then even ones !
- Non-redundant generation



Worst-case: Divide evenly at each step [GK81] $\Rightarrow \Theta(n \log(n))$

Message #3

Boustrophedon search saves $\Theta(\frac{n}{\log n})/\Omega(\frac{\sqrt{n}}{\log n})$ worst/average case.

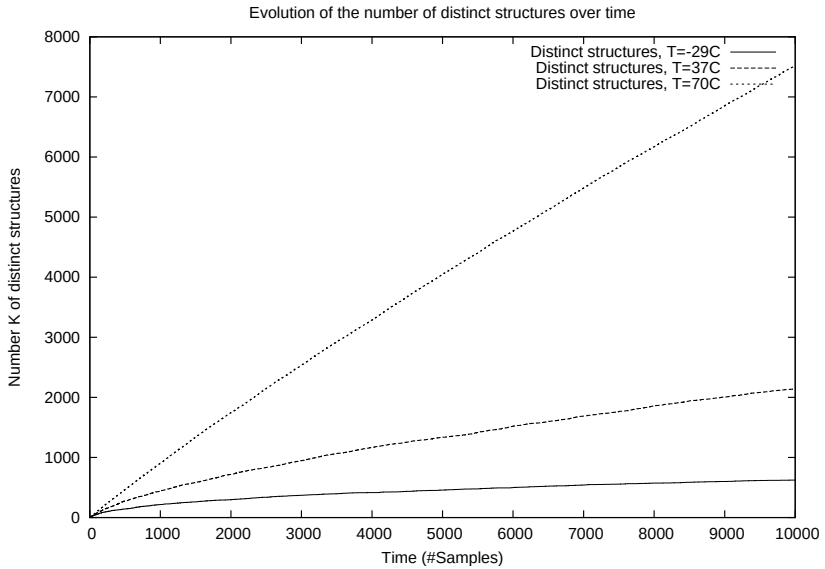
Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)

For each sampled structure one can compute the actual probability

$\Rightarrow$ It does not make any sense to sample it twice!



How long will it take to get k **distinct** samples?

Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)

  How long will it take to get k distinct samples?

Full collection (k = #structures): $E[C] \approx \mathcal{Z}'.n$
Way larger than #structures $\Rightarrow$ Exponential number of collisions.

Numerical values (Homopolymer/Nussinov energy/T=37):

$$E[C] \sim K \cdot 4.332^n/\sqrt{n} \text{ and \#structures: } S_n \sim K' \cdot 2.618^n/n\sqrt{n}$$

$\Rightarrow$ Each structure is sampled $1.65^n \cdot n$ times ($\neq \Theta(n)$ uniform dist.)

Message #4

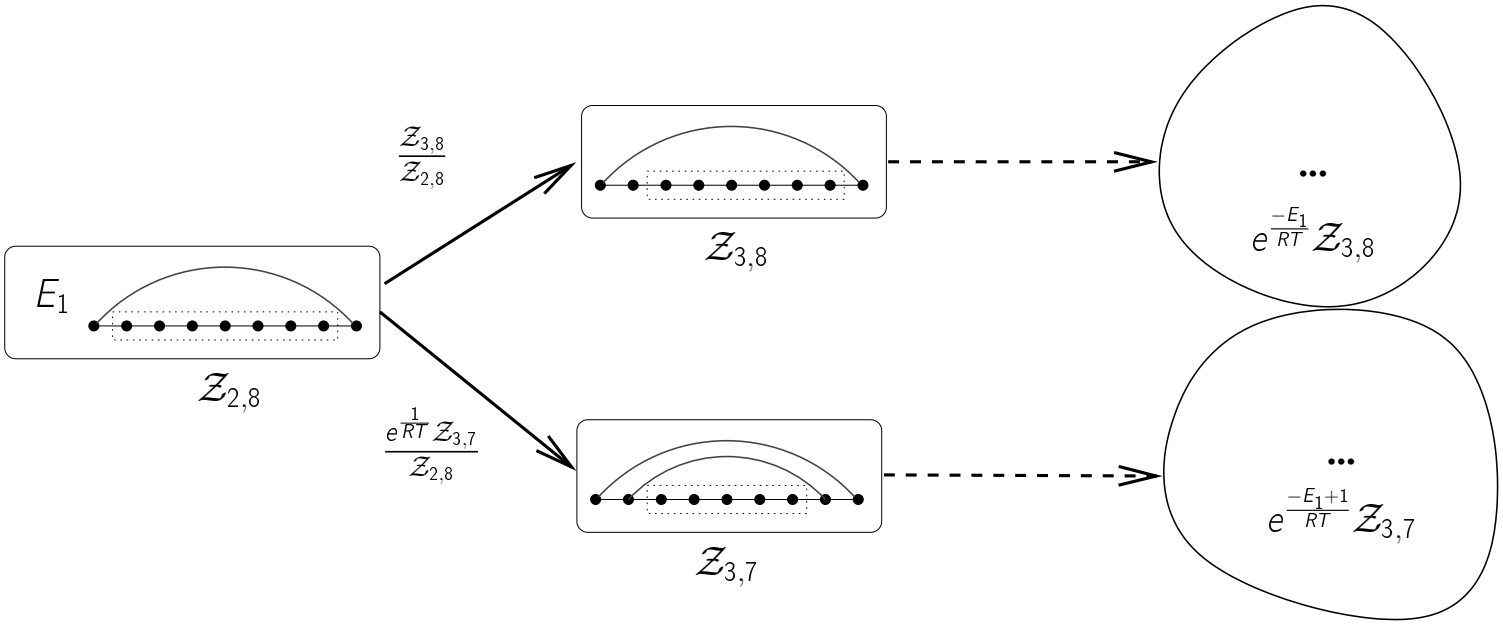
For any RNA there exists k such that the time for sampling k **distinct** sec. str. is **heavily dominated** by the cost of collisions.

k depends on the length $\Rightarrow$ Still need to push further our analysis...

Efficient statistical sampling

How to improve statistical sampling?

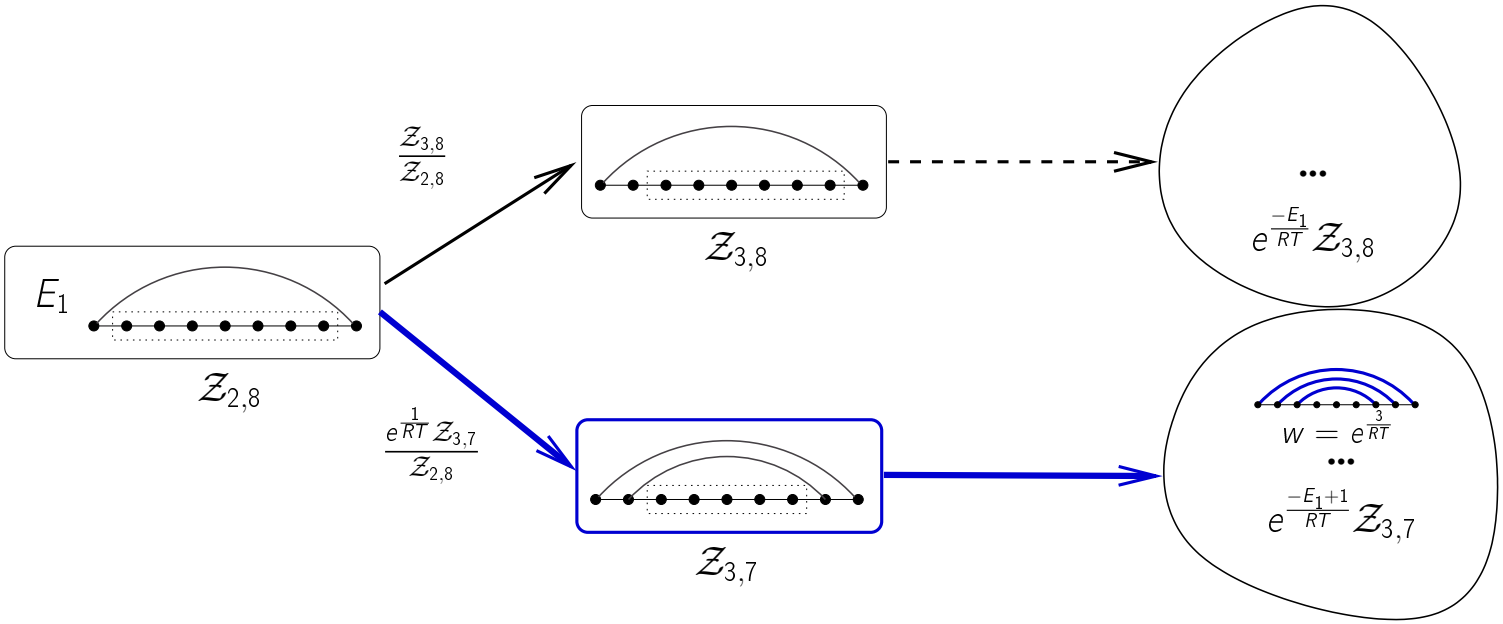
- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)
  - Build prefix tree for parse traces, storing in each node the contributions $K = \sum_{S \in \mathcal{R}} e^{\frac{-E_S}{RT}}$ of already sampled structures $\mathcal{R}$
  - During traceback, modify contributions of terms using K [Pon08]



Efficient statistical sampling

How to improve statistical sampling?

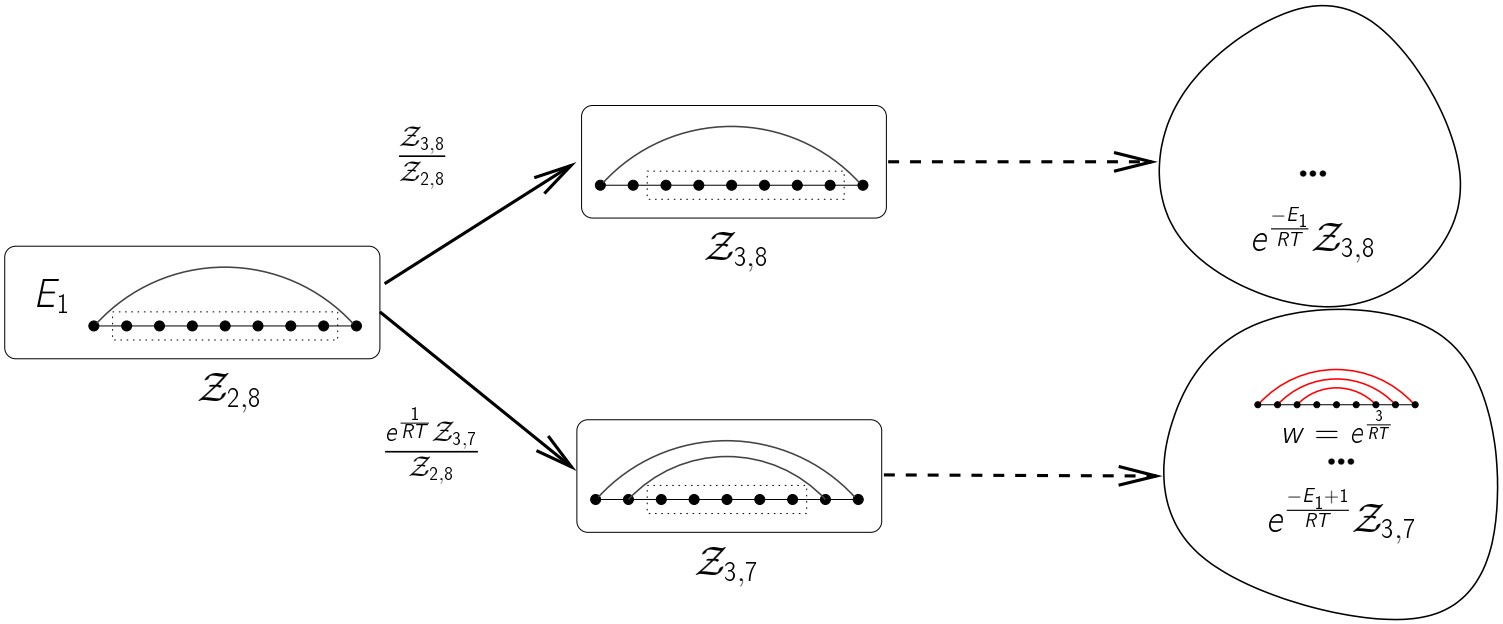
- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)
  - Build prefix tree for parse traces, storing in each node the contributions $K = \sum_{S \in \mathcal{R}} e^{\frac{-E_S}{RT}}$ of already sampled structures $\mathcal{R}$
  - During traceback, modify contributions of terms using K [Pon08]



Efficient statistical sampling

How to improve statistical sampling?

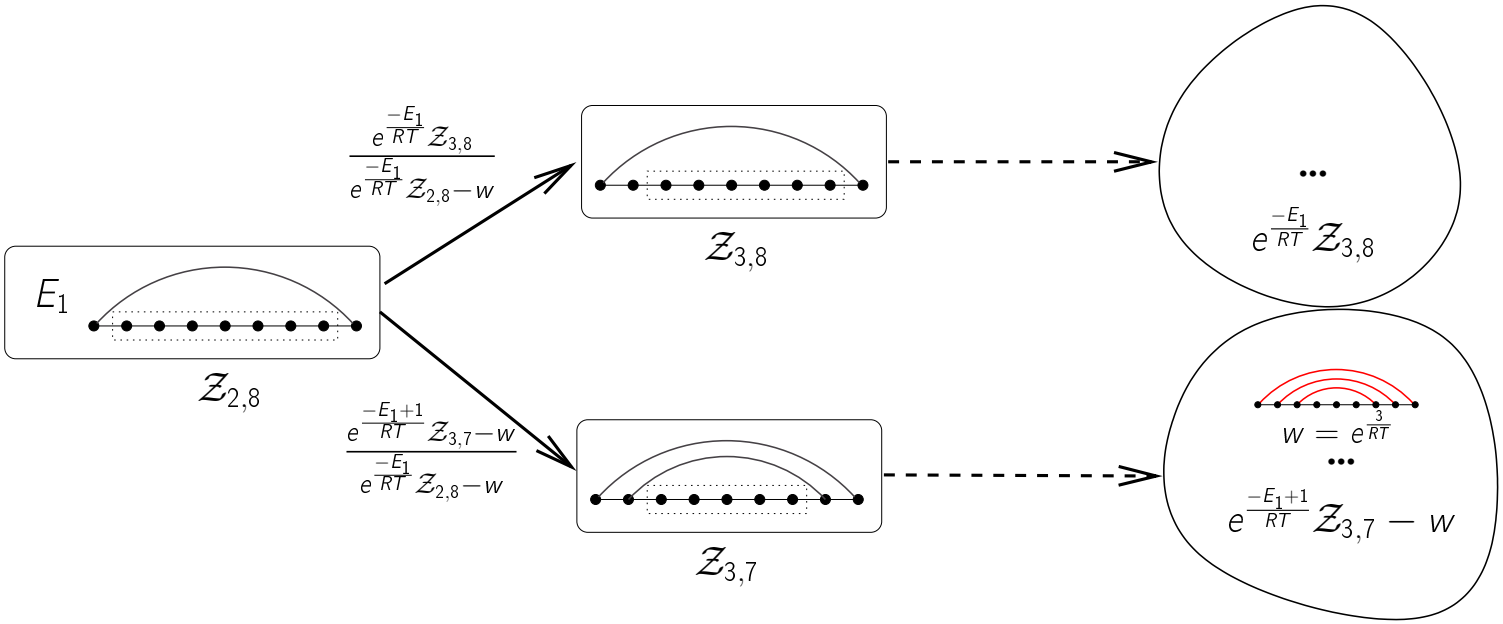
- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)
  - Build prefix tree for parse traces, storing in each node the contributions $K = \sum_{S \in \mathcal{R}} e^{\frac{-E_S}{RT}}$ of already sampled structures $\mathcal{R}$
  - During traceback, modify contributions of terms using K [Pon08]



Efficient statistical sampling

How to improve statistical sampling?

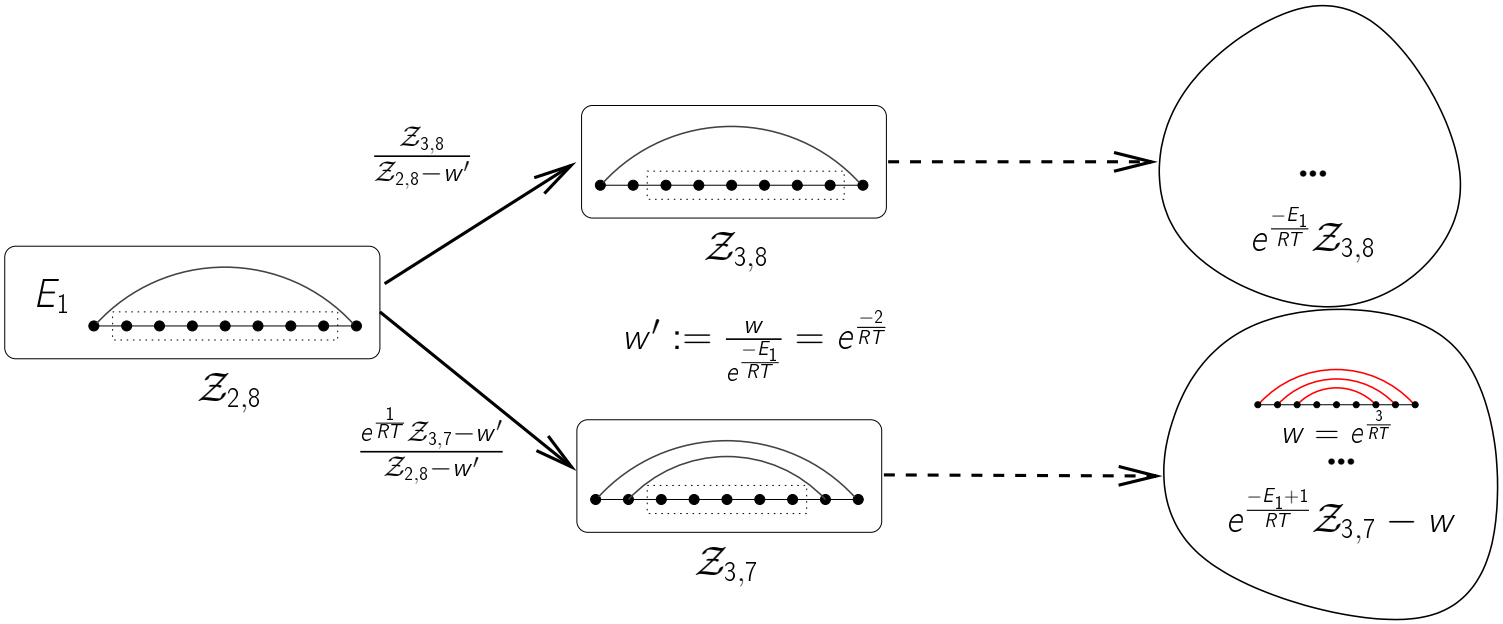
- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)
  - Build prefix tree for parse traces, storing in each node the contributions $K = \sum_{S \in \mathcal{R}} e^{\frac{-E_S}{RT}}$ of already sampled structures $\mathcal{R}$
  - During traceback, modify contributions of terms using K [Pon08]



Efficient statistical sampling

How to improve statistical sampling?

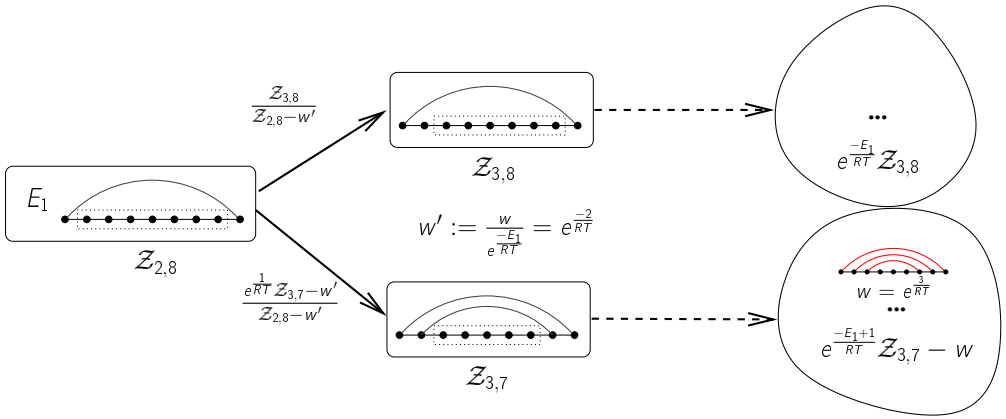
- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)
  - Build prefix tree for parse traces, storing in each node the contributions $K = \sum_{S \in \mathcal{R}} e^{\frac{-E_S}{RT}}$ of already sampled structures $\mathcal{R}$
  - During traceback, modify contributions of terms using K [Pon08]



Efficient statistical sampling

How to improve statistical sampling?

- Improve time complexity
- Non-redundant generation (with D. Gardy and A. Lorenz)
  - Build prefix tree for parse traces, storing in each node the contributions $K = \sum_{S \in \mathcal{R}} e^{\frac{-E_S}{RT}}$ of already sampled structures $\mathcal{R}$
  - During traceback, modify contributions of terms using K [Pon08]



Message #5

Storing parse trees and biasing local choices, one can perform non-redundant sampling in $\mathcal{O}(kn\log(n))$ time.

Conclusions

- Combinatorics gives a convenient framework for validating/analyzing/improving dynamic programming algorithms
- Statistical sampling = (Weighted) random generation of combinatorial structures + constraints
- During stochastic traceback, reordering comparisons saves time!
- One does not benefit from redundancy $\Rightarrow$ Non-redundant sampling

Open questions:

- When do collisions overcome the complexity of sampling?
- Does there exist sequential alternatives to RNASubopt?

Thanks for E. Rivas and E. Westhof!!!

References I

Y. Ding and E. Lawrence.

A statistical sampling algorithm for RNA secondary structure prediction.
Nucleic Acids Research, 31(24):7280–7301, 2003.

A. Denise, O. Roques, and M. Termier.

Random generation of words of context-free languages according to the frequencies of letters.
In D. Gardy and A. Mokkadem, editors, *Mathematics and Computer Science: Algorithms, Trees, Combinatorics and probabilities*, Trends in Mathematics, pages 113–125. Birkhaüser, 2000.

P. Flajolet, P. Zimmermann, and B. Van Cutsem.

Calculus for the random generation of labelled combinatorial structures.
Theoretical Computer Science, 132:1–35, 1994.

D. H. Greene and D. E. Knuth.

Mathematics for the Analysis of Algorithms.
Birkhauser Boston, 1981.

J.S. McCaskill.

The equilibrium partition function and base pair binding probabilities for RNA secondary structure.
Biopolymers, 29:1105–1119, 1990.

Y. Ponty.

Efficient sampling of RNA secondary structures from the boltzmann ensemble of low-energy: The boustrophedon method.
Journal of Mathematical Biology, 56(1-2):107–127, Jan 2008.

M. S. Waterman.

Secondary structure of single stranded nucleic acids.
Advances in Mathematics Supplementary Studies, 1(1):167–212, 1978.