Supplementary Material for "Stochastic Sampling of the RNA Structural Alignment Space,"

Arif Ozgun Harmanci[a], Gaurav Sharma[a,c], David H. Mathews[b,c,*]

[a]*Department of Electrical and Computer Engineering, University of Rochester, Hopeman 204, RC Box 270126, Rochester, NY 14627, USA*

[b]*Department of Biochemistry and Biophysics, University of Rochester Medical Center, 601 Elmwood Avenue, Box 712, Rochester, NY 14642, USA*

[c]*Department of Biostatistics and Computational Biology, University of Rochester Medical Center, 601 Elmwood Avenue, Box 630, Rochester, NY 14642, USA*

Overview

The algorithmic details of stochastic sampling method are presented. The notations are introduced first. Then the relationship between the partition function arrays in the PARTS algorithm and the Structural Alignment Atoms (SAAs) are presented. The details and flow of iterative sampling are presented next followed by the recursions for iterative sampling utilizing the partition function arrays. The last section presents the computation of the Calinski-Harabasz index and centroid structures and alignments.

Notation

The two input sequences are denoted by $\mathbf{x}_1$ and $\mathbf{x}_2$ and their lengths by N_1 and N_2, respectively. Throughout the description of the recursions, i and j denote nucleotide indices in the first sequence and k and l denote nucleotide indices in the second sequence. $\mathbf{S}_1$ and $\mathbf{S}_2$ are used to denote secondary structures of $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. $\mathbf{A}$ denotes a sequence alignment between $\mathbf{x}_1$ and $\mathbf{x}_2$. An SAA at sequence indices $i, j \in \mathbf{x}_1$ and $k, l \in \mathbf{x}_2$ is denoted by $\chi(i, j, k, l)$. The partition function arrays are denoted by ψ where an array location is indexed by a quadruple (i, j, k, l) where i and j are indices in $\mathbf{x}_1$ and k and l are indices in $\mathbf{x}_2$.

Relationship between Partition Function Arrays and SAAs

The partition function computation in PARTS utilizes 8 arrays such that an array location $\psi(i, j, k, l)$ stores the summation of exponentials of negative pseudo free energies of structural alignments between sequence fragments from index i to j in $\mathbf{x}_1$ and from index k to l in $\mathbf{x}_2$ under constraints on the types of structural alignments that a given array ψ handles. The set of arrays, ψ, was defined previously in [1] and is briefly summarized here. For an array ψ and nucleotide indices i, j, k, l possibilities of base pairing and alignment of nucleotides at indices $i, j \in \mathbf{x}_1$ and $k, l \in \mathbf{x}_2$ are defined by a SAA $\chi(i, j, k, l)$ that is specific to ψ. The set of possible SAAs $\chi(i, j, k, l)$ corresponding to each partition function array ψ at indices i, j, k, l are listed below:

1. V(i, j, k, l): Paired nucleotides at (i, j) are aligned to paired nucleotides at (k, l).
2. WL(i, j, k, l):
    (a) SAAs specified by V(i, j, k, l) array.

*Corresponding Author.

Email addresses: arharman@ece.rochester.edu (Arif Ozgun Harmanci), gaurav.sharma@rochester.edu (Gaurav Sharma), David_Mathews@urmc.rochester.edu (David H. Mathews).

   (b) Unpaired nucleotide i is aligned to unpaired nucleotide at k
   (c) Unpaired nucleotide i is inserted
   (d) Unpaired nucleotide k is inserted
   (e) Unpaired nucleotide j is aligned to unpaired nucleotide at l
   (f) Unpaired nucleotide j is inserted
   (g) Unpaired nucleotide l is inserted

3. Vbpi(i, j, k, l):
   (a) Base pair nucleotides at (i, j) are inserted
   (b) Base pair nucleotides at (k, l) are inserted

4. Vbau(i, j, k, l):
   (a) Paired nucleotides at (i, j) are aligned to unpaired nucleotides at k, l, respectively.
   (b) Paired nucleotides at (k, l) are aligned to unpaired nucleotides at i, j, respectively.

5. Wbau(i, j, k, l):
   (a) Paired nucleotides at (i, j) are aligned to unpaired nucleotides at k, l, respectively.
   (b) Paired nucleotides at (k, l) are aligned to unpaired nucleotides at i, j, respectively.

6. Wbpi(i, j, k, l):
   (a) Base pair nucleotides at (i, j) are inserted
   (b) Base pair nucleotides at (k, l) are inserted

7. SS(i, j, k, l):
   (a) Unpaired nucleotide i is aligned to unpaired nucleotide at k
   (b) Unpaired nucleotide i is inserted
   (c) Unpaired nucleotide k is inserted
   (d) Unpaired nucleotide j is aligned to unpaired nucleotide at l
   (e) Unpaired nucleotide j is inserted
   (f) Unpaired nucleotide l is inserted

WMBL, WMB, Vmhe, and Wmhi arrays are computed by processing of above arrays [1] and they do not directly specify SAAs at indices i, j, k, l.

Iterative Sampling of SAAs utilizing Partition Function Arrays

Iterative sampling algorithm is implemented via a *recursive stochastic traceback* of partition function arrays computed by PARTS algorithm [1]. Recursive stochastic traceback utilizes the dependencies in partition function arrays to compute $P(\chi(i, j, k, l)|\mathcal{S}_{\text{ext}}(i, j, k, l))$. This algorithm is similar to maximum *a posteriori* (MAP) traceback of PARTS algorithm with two main differences: 1) Stochastic traceback utilizes the partition function arrays whereas MAP traceback utilizes MAP arrays. 2) The stochastic traceback algorithm probabilistically generates a different structural alignment every time it operates on two RNA sequences. MAP traceback, however, always computes the same MAP structural alignment.

Recursions for Stochastic Traceback

A random selection operator, denoted by RandChoose($\mathrm{E}_1, \mathrm{E}_2, \cdots, \mathrm{E}_N$), is utilized in the stochastic traceback recursions to probabilistically chooses an index out of $[1, 2, \ldots, N]$ where the probability that an index i is sampled is proportional to the magnitude of E_i. The probability of choosing index i is therefore:

$$P(\mathrm{RandChoose}(\mathrm{E}_1, \mathrm{E}_2, \cdots, \mathrm{E}_N) = i) = \frac{E_i}{\sum_k E_k} \tag{1}$$

Algorithm 1 shows the algorithm for the RandChoose operator. The pseudo random number generation in Algorithm 1 is accomplished by *ran3* algorithm [2]. At the beginning of generation of a sample of structural alignments, ran3 is seeded with the number of seconds since the epoch as returned by *time* function. The number of seconds provides a constantly changing seed value for the pseudo random number generator, which ensures that the consecutively generated sample of structural alignments are not identical.

```
Generate a pseudo random number $\alpha \in [0, 1]$ ;
Set $\alpha = \alpha \times \sum_k E_k$ ;
Compute $C_i = \sum_1^i E_k$ and set $C_0 = 0$ ;
Find index $i$ such that $C_{i-1} \leq \alpha \leq C_i$; ;
return $i$ ;
```

Algorithm 1: RandChoose Algorithm

```
Set $s = \mathrm{RandChoose}(\mathrm{W}(1, N_1, 1, N_2), \mathrm{WMB}(1, N_1, 1, N_2))$ ;
if $s = 1$ then
    PUSH $\{1, N_1, 1, N_2, \mathrm{W}\}$ ON STACK;
else
    PUSH $\{1, N_1, 1, N_2, \mathrm{WMB}\}$ ON STACK;
while STACK NOT EMPTY do
    POP $\{i, j, k, l,$ ARRAY_ID$\}$;
    TRACEBACK $\{i, j, k, l,$ ARRAY_ID$\}$;
```

Algorithm 2: Main Stochastic Traceback Loop

Algorithm 2 shows the main stochastic traceback loop. Stochastic traceback utilizes a stack to ensure correct traceback of multibranched structures. A structural alignment is built by recursively backtracking the partition function arrays starting with $\mathrm{W}(1, N_1, 1, N_2)$ and $\mathrm{WMB}(1, N_1, 1, N_2)$. At each recursion, $\{i, j, k, l, \psi\}$ is popped from stack and used to initiate a stochastic traceback in accordance with partition function recursions given in [1]. The structures $\mathbf{S}_1$, $\mathbf{S}_2$ and sequence alignment $\mathbf{A}$ are updated based on the constraints that $\psi(i, j, k, l)$ imposes on pairing and alignment of nucleotides at indices i, j, k, l. The stochastic traceback for each partition function array $\{i, j, k, l, \psi\}$ is listed below:

- $\{i, j, k, l, \psi\} = \{i, j, k, l, \mathrm{V}\}$

```
// Set appropriate structure and alignment elements ;
ADD $(i,j)$ to $\mathbf{S}_1$, $(k,l)$ to $\mathbf{S}_2$, $\{(i,k,\mathrm{ALN}),(j,l,\mathrm{ALN})\}$ to $A$;
/* Sample one of 5 components from decomposition of V$(i,j,k,l)$ as described in [1].  V$(i,j,k,l)$
   has 5 components in total:  W, WMB, $\mathrm{Wmhi}_1$, $\mathrm{Wmhi}_2$, and SS                              */
$s = \mathrm{RandChoose}(\mathrm{W}(i+1,j-1,k+1,l-1)$, // Indexed with $i=1$
$\mathrm{WMB}(i+1,j-1,k+1,l-1)$, // Indexed with $i=2$
$\mathrm{Wmhi}_1(i+1,j-1,k+1,l-1)$, // Indexed with $i=3$
$\mathrm{Wmhi}_2(i+1,j-1,k+1,l-1)$, // Indexed with $i=4$
$\mathrm{SS}(i+1,j-1,k+1,l-1))$ ; // Indexed with $i=5$
// Resolve which component of V$(i,j,k,l)$ is sampled and push it on stack ;
switch $s$ do
    case 1
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{W}\}$ ON STACK;
    case 2
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{WMB}\}$ ON STACK;
    case 3
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{Wmhi}_1\}$ ON STACK;
    case 4
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{Wmhi}_2\}$ ON STACK;
    case 5
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{SS}\}$ ON STACK;
```

Algorithm 3: Algorithm for Stochastic Traceback of $\mathrm{V}(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{SS}\}$

```
Set $s = \mathrm{RandChoose}(\pi_{u_1}(i+1)\ \pi_a(i+1,k,\mathrm{INS1})\ \mathrm{SS}(i+1,j,k,l)$
$\pi_{u_2}(k+1)\ \pi_a(i,k+1,\mathrm{INS2})\ \mathrm{SS}(i,j,k+1,l)$,
$\pi_{u_1}(i+1)\ \pi_{u_2}(k+1)\ \pi_a(i+1,k+1,\mathrm{ALN})\ \mathrm{SS}(i+1,j,k+1,l))$ ;
switch $s$ do
    case 1
        ADD $(i+1,k,\ \mathrm{INS1})$ to $A$;
        PUSH $\{i+1,j,k,l,\mathrm{SS}\}$;
    case 2
        ADD $(i,k+1,\ \mathrm{INS2})$ to $A$;
        PUSH $\{i+1,j,k,l,\mathrm{SS}\}$;
    case 3
        ADD $(i+1,k+1,\ \mathrm{ALN})$ to $A$;
        PUSH $\{i+1,j,k+1,l,\mathrm{SS}\}$;
```

Algorithm 4: Stochastic Backtrack Algorithm for $\mathrm{SS}(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{WL}\}$

```
$k_1 = \pi_{u_1}(i)\ \pi_{u_2}(k)\ \pi_a(i,k,\mathrm{ALN})$;
$k_2 = \pi_{u_1}(i)\ \pi_a(i,k-1,\mathrm{INS1})$;
$k_3 = \pi_{u_2}(k)\ \pi_a(i-1,k,\mathrm{INS2})$;
$s = \mathrm{RandChoose}(\mathrm{V}(i,j,k,l)$,
$\mathrm{Vmhe}(i,j,k,l)$, $k_1 \times \mathrm{WL}(i+1,j,k+1,l)$,
$k_2 \times \mathrm{WL}(i+1,j,k,l)$,
$k_3 \times \mathrm{WL}(i,j,k+1,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{V}\}$;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Vmhe}\}$;
    case 3
        ADD $(i,k,$ ALN$)$ to $A$;
        PUSH $\{i+1,j,k+1,l,\mathrm{WL}\}$;
    case 4
        ADD $(i,k-1,$ INS1$)$ to $A$;
        PUSH $\{i+1,j,k,l,\mathrm{WL}\}$;
    case 5
        ADD $(i-1,k,$ INS2$)$ to $A$;
        PUSH $\{i,j,k+1,l,\mathrm{WL}\}$;
```
Algorithm 5: Stochastic Backtrack Algorithm for $\mathrm{WL}(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{W}\}$

```
$k_1 = \pi_{u_1}(j)\ \pi_{u_2}(l)\ \pi_a(j,l,\mathrm{ALN})$;
$k_2 = \pi_{u_1}(j)\ \pi_a(j,l,\mathrm{INS1})$;
$k_3 = \pi_{u_2}(l)\ \pi_a(j,l,\mathrm{INS2})$;
$s = \mathrm{RandChoose}(\mathrm{WL}(i,j,k,l)$,
$k_1\ \mathrm{W}(i,j-1,k,l-1)$,
$k_2\ \mathrm{W}(i,j-1,k,l)$,
$k_3\ \mathrm{W}(i,j,k,l-1))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{WL}\}$;
    case 2
        ADD $(j,l,$ ALN$)$ to $A$;
        PUSH $\{i,j-1,k,l-1,\mathrm{W}\}$;
    case 3
        ADD $(j,l,$ INS1$)$ to $A$;
        PUSH $\{i,j-1,k,l,\mathrm{W}\}$;
    case 4
        ADD $(j,l,$ INS2$)$ to $A$;
        PUSH $\{i,j,k,l-1,\mathrm{W}\}$;
```
Algorithm 6: Stochastic Backtrack Algorithm for $\mathrm{W}(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{WMBL}\}$

```
/* Components of WMBL$(i,j,k,l)$ are all possible concatenations of WL and WMBL arrays in the
   form of $\mathrm{WL}(i,i_p,k,k_p)\times(\mathrm{WL}(i_p+1,j,k_p+1,l)+\mathrm{WMBL}(i_p+1,j,k_p+1,l))$,  $i<ip<j$,  $k<k_p<l$  */
$\mathrm{WMBLComponents}[] = \{\mathrm{WL}(i,i+1,k,k+1)\times(\mathrm{WL}(i+2,j,k+2,l)+\mathrm{WMBL}(i+2,j,k+2,l)),$
$\mathrm{WL}(i,i+3,k,k+3)\times(\mathrm{WL}(i+4,j,k+4,l)+\mathrm{WMBL}(i+4,j,k+4,l)),\cdots,$
$\mathrm{WL}(i,i_p,k,k_p)\times(\mathrm{WL}(i_p+1,j,k_p+1,l)+\mathrm{WMBL}(i_p+1,j,k_p+1,l)),\cdots,$
$\mathrm{WL}(i,j-2,k,l-2)\times(\mathrm{WL}(j-1,j,l-1,l)+\mathrm{WMBL}(j-1,j,l-1,l)\}$ ;
// Sample from WMBLComponents array. $s = \mathrm{RandChoose}(\mathrm{WMBLComponents})$ ;
Find $i_p$ and $k_p$ such that
$\mathrm{WL}(i,i_p,k,k_p)\times(\mathrm{WL}(i_p+1,j,k_p+1,l)+\mathrm{WMBL}(i_p+1,j,k_p+1,l)) = \mathrm{WMBLComponents}[s]$ ;
// Push WL component
PUSH $(i,i_p,k,k_p,\mathrm{WL})$;
// Sample again for WL$(i_p+1,j,k_p+1,l)$ and WMBL$(i_p+1,j,k_p+1,l)$
$s = \mathrm{RandChoose}(\mathrm{WL}(i_p+1,j,k_p+1,l),\mathrm{WMBL}(i_p+1,j,k_p+1,l))$
switch $s$ do
    case 1
        PUSH $\{i_p+1,j,k_p+1,l,\mathrm{WL}\}$;
    case 2
        PUSH $\{i_p+1,j,k_p+1,l,\mathrm{WMBL}\}$;
```

Algorithm 7: Stochastic Backtrack Algorithm for WMBL(i,j,k,l)

- $\{i,j,k,l,\,\psi\} = \{i,j,k,l,\,\mathrm{WMB}\}$

```
$k_1 = \pi_{u_1}(j)\ \ \pi_{u_2}(l)\ \ \pi_a(j,l,\mathrm{ALN})$;
$k_2 = \pi_{u_1}(j)\ \ \pi_a(j,l,\mathrm{INS1})$;
$k_3 = \pi_{u_2}(k)\ \ \pi_a(j,l,\mathrm{INS2})$;
$s = \mathrm{RandChoose}(\mathrm{WMBL}(i,j,k,l),$
$k_1\times\mathrm{WMB}(i,j-1,k,l-1),$
$k_2\times\mathrm{WMB}(i,j-1,k,l),$
$k_3\times\mathrm{WMB}(i,j,k,l-1))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{WMBL}\}$ ;
    case 2
    ADD $(j,l,\mathrm{ALN})$ to $A$ ;
    PUSH $\{i,j-1,k,l-1,\mathrm{WMB}\}$ ;
    case 3
        ADD $(j,l,\mathrm{INS1})$ to $A$;
        PUSH $\{i,j-1,k,l,\mathrm{WMB}\}$ ;
    case 4
        ADD $(j,l,\mathrm{INS2})$ to $A$;
        PUSH $\{i,j,k,l-1,\mathrm{WMB}\}$ ;
```

Algorithm 8: Stochastic Backtrack Algorithm for WMB(i,j,k,l)

- $\{i,j,k,l,\,\psi\} = \{i,j,k,l,\,\mathrm{Vbpi}_1\}$

```
$k = \pi_{p_1}(i,j)\ \ \pi_a(i,k-1,\mathrm{INS1})\ \ \pi_a(j,l,\mathrm{INS1})$ ;
ADD $(i,j)$ to $\mathbf{S}_1$;
ADD $(i,k-1,$ INS1$)$ to $A$;
ADD $(j,l,$ INS1$)$ to $A$;
$s = \mathrm{RandChoose}(k \times \mathrm{Vmhe}(i+1,j-1,k,l),$
$k \times \mathrm{V}(i+1,j-1,k,l),$
$k \times \mathrm{Wmhi}_2(i+1,j-1,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i+1,j-1,k,l,\mathrm{Vmhe}\}$;
    case 2
        PUSH $\{i+1,j-1,k,l,\mathrm{V}\}$;
    case 3
        PUSH $\{i+1,j-1,k,l,\mathrm{Wmhi}_2\}$;
```

Algorithm 9: Stochastic Backtrack Algorithm for $\mathrm{Vpi}_1(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Vbpi}_2\}$

```
$k = \pi_{p_2}(k,l)\ \ \pi_a(i-1,k,\mathrm{INS2})\ \ \pi_a(j,l,\mathrm{INS2})$;
ADD $(k,l)$ to $\mathbf{S}_2$;
ADD $(i-1,k,$ INS2$)$ to $A$;
ADD $(j,l,$ INS2$)$ to $A$;
$s = \mathrm{RandChoose}(k \times \mathrm{Vmhe}(i,j,k+1,l-1),$
$k \times \mathrm{V}(i,j,k+1,l-1),$
$k \times \mathrm{Wmhi}_2(i,j,k+1,l-1))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k+1,l-1,\mathrm{Vmhe}\}$;
    case 2
        PUSH $\{i,j,k+1,l-1,\mathrm{V}\}$;
    case 3
        PUSH $\{i,j,k+1,l-1,\mathrm{Wmhi}_2\}$;
```

Algorithm 10: Stochastic Backtrack Algorithm for $\mathrm{Vbpi}_2(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Wbpi}_1\}$

```
ADD $(i, j)$ to $\mathbf{S}_1$;
ADD $(i, k-1, \text{INS1})$ to $A$;
ADD $(j, l, \text{INS1})$ to $A$;
ADD $(k, l)$ to $\mathbf{S}_1$ ;
$k = \pi_{p_1}(i, j)\ \ \pi_a(i, k-1, \text{INS1})\ \ \pi_a(j, l, \text{INS1})$
$\text{OpenWScore} = \text{W}(i+1, j-1, k, l) - \text{V}(i+1, j-1, k, l) - \text{Vmhe}(i+1, j-1, k, l)$
$s = \text{RandChoose}(k \times \text{OpenWScore},$
$k \times \text{WMB}(i+1, j-1, k, l),$
$k \times \text{Wmhi}_1(i+1, j-1, k, l),$
$k \times \text{SS}_1(i+1, j-1, k, l))$ ;
switch $s$ do
    case 1
        PUSH $\{i+1, j-1, k, l, \text{OpenW}\}$;
    case 2
        PUSH $\{i+1, j-1, k, l, \text{Wmhi}_2\}$;
    case 3
        PUSH $\{i+1, j-1, k, l, \text{Wmhi}_1\}$;
    case 4
        PUSH $\{i+1, j-1, k, l, \text{SS}_1\}$;
```
Algorithm 11: Stochastic Backtrack Algorithm for $\text{Wbpi}_1(i, j, k, l)$

- $\{i, j, k, l,\ \psi\} = \{i, j, k, l,\ \text{Wbpi}_2\}$

```
ADD $(k, l)$ to $\mathbf{S}_2$;
ADD $(i-1, k, \text{INS2})$ to $A$;
ADD $(j, l, \text{INS2})$ to $A$;
ADD $(k, l)$ to $\mathbf{S}_2$ ;
$k = \pi_{p_2}(k, l)\ \ \pi_a(i-1, k, \text{INS2})\ \ \pi_a(j, l, \text{INS2})$
$\text{OpenWScore} = \text{W}(i, j, k+1, l-1) - \text{V}(i, j, k+1, l-1) - \text{Vmhe}(i, j, k+1, l-1)$
$s = \text{RandChoose}(k \times \text{OpenWScore},$
$k \times \text{WMB}(i, j, k+1, l-1),$
$k \times \text{Wmhi}_1(i, j, k+1, l-1),$
$k \times \text{SS}(i, j, k+1, l-1))$ ;
switch $s$ do
    case 1
        PUSH $\{i, j, k+1, l-1, \text{OpenW}\}$;
    case 2
        PUSH $\{i, j, k+1, l-1, \text{Wmhi}_2\}$;
    case 3
        PUSH $\{i, j, k+1, l-1, \text{Wmhi}_1\}$;
    case 4
        PUSH $\{i, j, k+1, l-1, \text{SS}_1\}$;
```
Algorithm 12: Stochastic Backtrack Algorithm for $\text{Wbpi}_2(i, j, k, l)$

- $\{i, j, k, l,\ \psi\} = \{i, j, k, l,\ \text{OpenW}\}$

```
$k_1 = \pi_{u_1}(j)\ \pi_{u_2}(l)\ \pi_a(j,l,\mathrm{ALN})$;
$k_2 = \pi_{u_1}(j)\ \pi_a(j,l,\mathrm{INS1})$;
$k_3 = \pi_{u_2}(l)\ \pi_a(j,l,\mathrm{INS2})$;
$s = \mathrm{RandChoose}(\mathrm{WL}(i,j,k,l) - \mathrm{V}(i,j,k,l) - \mathrm{Vmhe}(i,j,k,l)$,
$k_1 \times \mathrm{W}(i,j-1,k,l-1)$,
$k_2 \times \mathrm{W}(i,j-1,k,l)$,
$k_3 \times \mathrm{W}(i,j,k,l-1))$ ;
**switch** $s$ **do**
    **case** 1
        PUSH $\{i,j,k,l,\mathrm{OpenWL}\}$;
    **case** 2
        ADD $(j,l,$ ALN$)$ to $A$;
        PUSH $\{i,j-1,k,l-1,\mathrm{W}\}$;
    **case** 3
        ADD $(j,l,$ INS1$)$ to $A$;
        PUSH $\{i,j-1,k,l,\mathrm{W}\}$;
    **case** 4
        ADD $(j,l,$ INS2$)$ to $A$;
        PUSH $\{i,j,k,l-1,\mathrm{W}\}$;
```

Algorithm 13: Stochastic Backtrack Algorithm for OpenW(i,j,k,l)

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,$ OpenWL$\}$

```
$k_1 = \pi_{u_1}(i) \times \pi_{u_2}(k)\ \pi_a(i,k,\mathrm{ALN})$;
$k_2 = \pi_{u_1}(i) \times \pi_a(i,k-1,\mathrm{INS1})$;
$k_3 = \pi_{u_2}(k) \times \pi_a(i-1,k,\mathrm{INS2})$;
$s = \mathrm{RandChoose}(k_1 \times \mathrm{WL}(i+1,j,k+1,l)$,
$k_2 \times \mathrm{WL}(i+1,j,k,l)$,
$k_3 \times \mathrm{WL}(i,j,k+1,l))$ ;
**switch** $s$ **do**
    **case** 1
        ADD $(i,k,$ ALN$)$ to $A$;
        PUSH $\{i+1,j,k+1,l,\mathrm{WL}\}$;
    **case** 2
        ADD $(i,k-1,$ INS1$)$ to $A$;
        PUSH $\{i+1,j,k,l,\mathrm{WL}\}$;
    **case** 3
        ADD $(i-1,k,$ INS2$)$ to $A$;
        PUSH $\{i,j,k+1,l,\mathrm{WL}\}$;
```

Algorithm 14: Stochastic Backtrack Algorithm for OpenWL(i,j,k,l)

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,$ Wbau$_1\}$

```
ADD $(i, j)$ to $\mathbf{S}_1$;
ADD $(i, k,$ ALN$)$ to $A$;
ADD $(j, l,$ ALN$)$ to $A$;
$k = \pi_{p_1}(i, j)\ \ \pi_{u_2}(k)\ \ \pi_{u_2}(l)\ \ \pi_a(i, k, \mathrm{ALN})\ \ \pi_a(j, l, \mathrm{ALN}))$;
$s = \mathrm{RandChoose}(k \times (\mathrm{W}(i+1, j-1, k+1, l-1) - \mathrm{V}(i+1, j-1, k+1, l-1) - \mathrm{Vmhe}(i+1, j-1, k+1, l-1)),$
$k \times \mathrm{WMB}(i+1, j-1, k+1, l-1),$
$k \times \mathrm{Wmhi}_1(i+1, j-1, k+1, l-1),$
$k \times \mathrm{SS}(i+1, j-1, k+1, l-1))$ ;
switch $s$ do
    case 1
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{Wopen}\}$
    case 2
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{WMB}\}$;
    case 3
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{Wmhi}_1\}$;
    case 4
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{SS}\}$;
```

Algorithm 15: Stochastic Backtrack Algorithm for $\mathrm{Wbau}_1(i, j, k, l)$

- $\{i, j, k, l,\ \psi\} = \{i, j, k, l,\ \mathrm{Wbau}_2\}$

```
ADD $(k, l)$ to $\mathbf{S}_2$;
ADD $(i, k,$ ALN$)$ to $A$;
ADD $(j, l,$ ALN$)$ to $A$;
$k = \pi_{p_2}(k, l)\ \ \pi_{u_1}(i)\ \ \pi_{u_1}(j)\ \ \pi_a(i, k, \mathrm{ALN})\ \ \pi_a(j, l, \mathrm{ALN})$;
$s = \mathrm{RandChoose}(k \times (\mathrm{W}(i+1, j-1, k+1, l-1) - \mathrm{V}(i+1, j-1, k+1, l-1) - \mathrm{Vmhe}(i+1, j-1, k+1, l-1)),$
$k \times \mathrm{WMB}(i+1, j-1, k+1, l-1),$
$k \times \mathrm{Wmhi}_1(i+1, j-1, k+1, l-1),$
$k \times \mathrm{SS}(i+1, j-1, k+1, l-1))$ ;
switch $s$ do
    case 1
        PUSH $\{i, j, k, l, \mathrm{Wopen}\}$
    case 2
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{WMB}\}$;
    case 3
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{Wmhi}_2\}$;
    case 4
        PUSH $\{i+1, j-1, k+1, l-1, \mathrm{SS}\}$;
```

Algorithm 16: Stochastic Backtrack Algorithm for $\mathrm{Wbau}_2(i, j, k, l)$

- $\{i, j, k, l,\ \psi\} = \{i, j, k, l,\ \mathrm{Vbau}_1\}$

```
ADD $(i,j)$ to $\mathbf{S}_1$;
ADD $(i,k,$ ALN$)$ to $A$;
ADD $(j,l,$ ALN$)$ to $A$;
$s = \mathrm{RandChoose}(k \times \mathrm{Vmhe}(i+1,j-1,k+1,l-1),$
$k \times \mathrm{V}(i+1,j-1,k+1,l-1),$
$k \times \mathrm{Wmhi}_2(i+1,j-1,k+1,l-1))$ ;
$k = \pi_{p_1}(i,j)\ \ \pi_{u_2}(k)\ \ \pi_{u_2}(l)\ \ \pi_a(i,k,\mathrm{ALN})\ \ \pi_a(j,l,\mathrm{ALN})$;
switch $s$ do
    case 1
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{Vmhe}\}$;
    case 2
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{V}\}$;
    case 3
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{Wmhi}_2\}$;
```

Algorithm 17: Stochastic Backtrack Algorithm for $\mathrm{Vbau}_1(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Vbau}_2\}$

```
ADD $(k,l)$ to $\mathbf{S}_2$;
ADD $(i,k,$ ALN$)$ to $A$;
ADD $(j,l,$ ALN$)$ to $A$;
$s = \mathrm{RandChoose}(k\ \ \mathrm{Vmhe}(i+1,j-1,k+1,l-1),$
$k \times \mathrm{V}(i+1,j-1,k+1,l-1),$
$k \times \mathrm{Wmhi}_1(i+1,j-1,k+1,l-1))$ ;
$k = \pi_{p_2}(k,l)\ \ \pi_{u_1}(i)\ \ \pi_{u_1}(j)\ \ \pi_a(i,k,\mathrm{ALN})\ \ \pi_a(j,l,\mathrm{ALN})$;
switch $s$ do
    case 1
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{Vmhe}\}$;
    case 2
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{V}\}$;
    case 3
        PUSH $\{i+1,j-1,k+1,l-1,\mathrm{Wmhi}_1\}$;
```

Algorithm 18: Stochastic Backtrack Algorithm for $\mathrm{Vbau}_2(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Wmhi}\}$

```
$s = \mathrm{RandChoose}(\mathrm{Wmhi}_1(i,j,k,l), \mathrm{Wmhi}_2(i,j,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{Wmhi}_1\}$ ;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Wmhi}_2\}$ ;
```

Algorithm 19: Stochastic Backtrack Algorithm for $\mathrm{Wmhi}(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Wmhi}_1\}$

```
$s = \mathrm{RandChoose}(\mathrm{Wbau}_1(i,j,k,l), \mathrm{Wbpi}_1(i,j,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{Wbau}_1\}$;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Wbpi}_1\}$;
```

Algorithm 20: Stochastic Backtrack Algorithm for $\mathrm{Wmhi}_2(i,j,k,l)$

```
$s = \mathrm{RandChoose}(\mathrm{Wbau}_2(i,j,k,l), \mathrm{Wbpi}_2(i,j,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{Wbau}_2\}$;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Wbpi}_2\}$;
```

Algorithm 21: Stochastic Backtrack Algorithm for $\mathrm{Wmhi}_2(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Vmhe}\}$

```
$s = \mathrm{RandChoose}(\mathrm{Vmhe}_1(i,j,k,l), \mathrm{Vmhe}_2(i,j,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{Vmhe}_1\}$;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Vmhe}_2\}$;
```

Algorithm 22: Stochastic Backtrack Algorithm for $\mathrm{Vmhe}(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Vmhe}_1\}$

```
$s = \mathrm{RandChoose}(\mathrm{Vbau}_1(i,j,k,l), \mathrm{Vbpi}_1(i,j,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{Vbau}_1\}$;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Vbpi}_1\}$;
```

Algorithm 23: Stochastic Backtrack Algorithm for $\mathrm{Vmhe}_1(i,j,k,l)$

- $\{i,j,k,l,\ \psi\} = \{i,j,k,l,\ \mathrm{Vmhe}_2\}$

```
$s = \mathrm{RandChoose}(\mathrm{Vbau}_2(i,j,k,l), \mathrm{Vbpi}_2(i,j,k,l))$ ;
switch $s$ do
    case 1
        PUSH $\{i,j,k,l,\mathrm{Vbau}_2\}$;
    case 2
        PUSH $\{i,j,k,l,\mathrm{Vbpi}_2\}$;
```

Algorithm 24: Stochastic Backtrack Algorithm for $\mathrm{Vmhe}_2(i,j,k,l)$

Computation of Calinski-Harabasz (CH) Index and Cluster Centroid Structure and Alignment

The stochastic sampling of the structural alignments yields a set of n representative structural alignments $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$, which in turn provide a corresponding set of structures $\mathbf{S}_{1,1}, \mathbf{S}_{1,2}, \ldots, \mathbf{S}_{1,n}$ for $\mathbf{x}_1$ and $\mathbf{S}_{2,1}, \mathbf{S}_{2,2}, \ldots, \mathbf{S}_{2,n}$ for $\mathbf{x}_2$ and a set of alignments $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$. Each sample of secondary structures or sequence alignments is hierarchically clustered using the *diana* algorithm [3]. This clustering process arranges the n sample elements (structures or alignments) in a hierarchical tree or dendrogram. For a hierarchical clustering tree of a sample of size n, the tree is cut at an appropriate height h_k to generate a set of k clusters denoted by $\mathbf{C}_k$. The m^{th} cluster in $\mathbf{C}_k$ is denoted by $\mathbf{C}_{km}$ for $1 \leq m \leq k$. $\mathbf{C}_{km}$ is the set of the sample indices of structures in m^{th} cluster such that $\mathbf{C}_{km}(i)$ is used to denote the sample index of i^{th} element in m^{th} cluster in $\mathbf{C}_k$. The number of elements in m^{th} cluster in $\mathbf{C}_k$ is denoted by n_m. The optimal cutting height $h_{k_{\text{opt}}}$ and a corresponding cluster count k_{opt} are determined to maximize the Calinski-Harabasz (CH) Index [4] defined as:

$$\text{CH}(k) = \frac{\left(\frac{\text{BGSS}(\mathbf{C}_k)}{k-1}\right)}{\left(\frac{\text{WGSS}(\mathbf{C}_k)}{n-1}\right)} \tag{2}$$

where BGSS stands for "between-groups sum of squares distance" and WGSS stands for "within-group sum of square distance." BGSS and WGSS are computed as:

$$\text{WGSS}(\mathbf{C}_k) = \sum_{m=1}^{k} (n_m - 1)\overline{d_m^2} \tag{3}$$

and

$$\text{BGSS}(\mathbf{C}_k) = (k-1)\overline{d^2} + \sum_{m=1}^{k} (n_m - 1)(\overline{d^2} - \overline{d_m^2}) \tag{4}$$

where $\overline{d^2}$ denotes the average of squared distances between all $n(n-1)/2$ pairs of elements and $\overline{d_m^2}$ denotes the average of squared distances between all $n_m(n_m-1)/2$ pairs of elements in m^{th} cluster in $\mathbf{C}_k$. $\overline{d^2}$ and $\overline{d_m^2}$ are computed as:

$$\overline{d^2} = \frac{2}{n(n-1)} \sum_{\substack{1 \leq a \leq n \\ a<b}} tr((\mathbf{M}_a - \mathbf{M}_b)(\mathbf{M}_a - \mathbf{M}_b)^{\text{T}}) \tag{5}$$

and

$$\overline{d_m^2} = \frac{2}{n_m(n_m-1)} \sum_{\substack{1 \leq a \leq n_m \\ a<b}} tr((\mathbf{M}_{\mathbf{C}_{km}(a)} - \mathbf{M}_{\mathbf{C}_{km}(b)})(\mathbf{M}_{\mathbf{C}_{km}(a)} - \mathbf{M}_{\mathbf{C}_{km}(b)})^{\text{T}}) \tag{6}$$

where $\mathbf{M}_i$ represents the matrix representation of element at sample index i, tr$(\ldots)$ denotes the trace of the matrix in the argument, and $(\ldots)^{\text{T}}$ represents the transpose of the matrix in the argument. For the clustering of secondary structures of an RNA sequence of length N, $\mathbf{M}_i$ is an upper triangular matrix of size $N \times N$ defined as:

$$\mathbf{M}_i(r,s) = \begin{cases} 1 & \textbf{if } r < s \text{ and nucleotides at indices } r \text{ and } s \text{ are paired in } i^{\text{th}} \text{ structure} \\ 0 & \textbf{otherwise} \end{cases} \tag{7}$$

For the clustering of sequence alignments of two RNA sequences of lengths N_1 and N_2, $\mathbf{M}_i$ is a matrix of size $N_1 \times N_2$ defined as:

$$\mathbf{M}_i(r,s) = \begin{cases} 1 & \textbf{if } \text{nucleotides at indices } r \text{ and } s \text{ are aligned in } i^{\text{th}} \text{ alignment} \\ 0 & \textbf{otherwise} \end{cases} \tag{8}$$

The term $tr((\mathbf{M}_{\mathbf{C}_{km}(a)} - \mathbf{M}_{\mathbf{C}_{km}(b)})(\mathbf{M}_{\mathbf{C}_{km}(a)} - \mathbf{M}_{\mathbf{C}_{km}(b)})^{\text{T}})$ in Equations (5) and (6) formulates the squared distance between elements a and b in sample. The squared distances is squared base pair distance and squared aligned position distance for structures and for alignments, respectively.

For the optimal number of clusters that maximize CH(k), the centroid structure of cluster $\mathbf{C}_{km}$ in C_k is the structure that has the smallest average base pair distance to all the structures in $\mathbf{C}_{km}$:

$$\mathbf{S}_{\text{cent}} = \underset{\mathbf{S}\in\mathbf{S}_{\text{all}}}{\operatorname{argmin}} \ \frac{1}{n_m}\sum_{a=1}^{n_m} \operatorname{tr}((\mathbf{M}_{\mathbf{S}} - \mathbf{M}_{\mathbf{C}_{km}(a)})(\mathbf{M}_{\mathbf{S}} - \mathbf{M}_{\mathbf{C}_{km}(a)})^{\mathrm{T}}) \tag{9}$$

where $\mathbf{S}_{\text{all}}$ represents the set of all possible secondary structures of the sequence, $\mathbf{S}_{\text{cent}}$ denotes the centroid structure, and $\mathbf{M}_{\mathbf{S}}$ is the upper triangular matrix representing the structure $\mathbf{S}$ as defined above by matrix representation of secondary structures. Two nucleotides at indices r, s are paired in the centroid structure of m^{th} cluster if more than $\lfloor\frac{n_m}{2}\rfloor$ of structures in the cluster have nucleotides at indices r, s paired. Because if more than $\lfloor\frac{n_m}{2}\rfloor$ of structures in the cluster have nucleotides at indices r, s paired, a structure with nucleotides at r, s paired has lower average distance to structures in the cluster than a structure that has nucleotides at r, s unpaired. Thus, the set of base pairs (r, s) in the centroid structure can be defined as:

$$\mathbf{S}_{\text{cent}} = \left\{(r, s) \ | \ \sum_{a=1}^{n_m} \mathbf{M}_{\mathbf{C}_{km}(a)}(r, s) > \lfloor\frac{n_m}{2}\rfloor\right\} \tag{10}$$

These computations are similar to the computation of centroid structures described by Ding et al. [5]. Computation of cluster centroid for sequence alignments can be similarly formulated.

References

[1] Harmanci, A. O., Sharma, G., and Mathews, D. H. (2008) PARTS: Probabilistic alignment for RNA joinT secondary structure prediction. *Nucleic Acids Res.,* **36**, 2406–2417.

[2] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992) *Numerical Recipes in C*, Cambridge University Press, Cambridge, U.K. second edition.

[3] Kaufman, L. and Rousseeuw, P. J. (2005) *Finding Groups in Data; An Introduction to Cluster Analysis*, Wiley, New York, NY.

[4] Calinski, R. and Harabasz, J. (1974) A dendrite method for cluster analysis. *Commun. Stat.,* **3**, 1 – 27.

[5] Ding, Y., Chan, C. Y., and Lawrence, C. E. (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA,* **11**, 1157–1166.