ABSTRACT
In this paper we present the idea of mining RNA families with structure histograms. A structure histogram is a histogram employing structures of some type as attributes. As structures of RNA sequences we adopt their secondary structures which are not pseudo-knots. Unfortunately to obtain the histogram for an RNA sequence of its length \( l \) needs more than the \((l/2)\)-th Catalan number time, but show that the value for every structure in the histogram is calculated in the time \( O(l^3) \). We also give some experimental results by applying structure histograms obtained from real RNA data to some mining methods and demonstrated the cases that structure histograms works effectively.

Keywords
RNA sequences, RNA families, Secondary structure, Data mining

1. INTRODUCTION
In this paper, we present an idea of mining RNA sequences with structure histograms. RNA is a kind of nucleic acid similar to DNA. In analyzing RNA molecules their structures are important because they are considered to affect the functions of the molecules. An RNA molecule forms a wind three-dimensional structure and the structure is based on its secondary structure, which is formed by the role of Watson-Crick complementary base pairs in it. In this paper we pay our attention to the secondary structures, and use them for mining of RNA data.

The secondary structure of a RNA molecule is obtained by representing it as a sequence, which we call an RNA sequence, consisting of four symbols A, U, C, and G, where U is used instead of T for DNA sequences. Pairs of A-U and C-G in the RNA sequence are called Watson-Crick complementary base pairs and some of them are used in forming the secondary structure. RNA sequences are stored in some database, e.g, Rfam database, and published through the Internet. Note that not all base pairs are used for secondary structure. This means that there can be theoretical secondary structures in an RNA sequence, and our idea is to use such theoretical ones as attributes in data mining. In this paper the word “secondary structure” includes such theoretical structures. Because secondary structures are formed with the base pairs, some of them can be formalized with a context free grammar and we use only such type of structures. A type of structures which cannot be represented with the grammar is “pseudo-knot”.

Our idea is to make histogram of structures for every RNA sequences. We intend to measure the strength of each structure in a given RNA sequences, by counting how many times the structure can occur as a theoretical structure in the sequences. In this paper we give a method of the number of occurrences of every structure. We also give some experimental results by applying structure histograms obtained from real RNA data to some mining methods.

This paper is organized as follows: In Section 2 we give our definitions of structures and their instances. In Section 3 we give the method of calculating the numbers of occurrences of every structure. In Section 4 we give some experimental results by using structure histograms, and we conclude in the last section.

2. STRUCTURES AND INSTANCES
We treat every RNA as a non-empty sequence of an alphabet \( \Sigma = \{A,U,C,G\} \), that is, an element of \( \Sigma^+ \) and call it an RNA-sequence. The length \( n \) of an RNA-sequence \( \sigma = a_1a_2\cdots a_n \) is denoted by \( n = |\sigma| \). A number \( i \) (\( 1 \leq i \leq n \))
is called an occurrence if it denotes the index of the symbol
$x_i$, and $σ[i]$ denotes the symbol $a_i$, and $σ[i:j]$ denotes the
continuous subsequence $σ = a_i a_{i+1} \cdots a_j$. A pair $(i, j)$ of
occurrences of $σ$ such that $i < j$ is called a candidate pair,
or $c$-pair for short, if $(σ[i], σ[j])$ is either of $(A, U)$, $(U, A)$,
$(C, G)$, and $(G, C)$. A $c$-pair represents the indices of $σ$ the
elements at which may construct a Watson-Crick complementar-
base pair.

For two $c$-pairs $p = (a_{i_1}, a_{j_1})$ and $q = (a_{i_2}, a_{j_2})$ in $σ$, we
define two binary relations $<$ and $\preceq$: $q < p$ if $i_2 < j_2 < i_1 < j_1$, and $q \preceq p$ if $i_1 < i_2 < j_2 < j_1$.

We define structures with a context free grammar $G$ on an
alphabet $X = \{x, y\}$ with a non-terminal $S$ and rules:

\begin{align*}
R_1 & : S \rightarrow xy, \\
R_2 & : S \rightarrow xSy, \\
R_3 & : S \rightarrow SS
\end{align*}

We call every word in $L(G)$ a structure. It is well-known that
the context free language $L(G)$ is, by replacing $x$ with ( and $y$ with ), the set of the sequences of parentheses correctly
matched, and the size of the set

$$C_n = \{w \in L(G) \mid |w| = 2n\}$$

is called the $n$-th Catalan number and given as

$$C_n = \frac{(2n)!}{n!n+1}. \quad (1)$$

In order to define matching of a structure $S$ and a sequence
$σ$, we distinguish $x$’s and $y$’s in $S$ in the following manner:
We give a suffix $k$ to each $x$ if the $x$ is the $k$-th occurrence in $S$
with scanning it from left to right. We also give a suffix $k$
to the $y$ which matches $x_k$. We regard every $x_k$ and $y_k$ as
a variable, and call the pair a variable pair. For example, a
structure $S = xxyzyxy$ is regarded as a sequence of variables
$x_1 x_2 y_2 y_1 x_3 y_3 y_1$.

A substitution for a structure $S = z_1 z_2 \cdots z_{2n}$ is a mapping
$θ : \{x_1, \ldots, x_n, y_1, \ldots, y_n\} \rightarrow Σ$ such that $(θ(x_k), θ(y_k))$ is
a $c$-pair for every $k = 1, 2, \ldots, n$. A sequence $σ$ in $Σ^+$ is an
instance of $S$ if $σ = Sθ$ for some substitution $θ$. The structure $S$
appears in a sequence $σ = a_1a_2\cdots a_n$ if $σ$ contains
an instance of $S$ as its subsequence, that is, there are indices
$i_1 < i_2 < \cdots < i_m$ and a substitution such that

$$a_{i_1} a_{i_2} \cdots a_{i_m} = Sθ$$

for some substitution $θ$. The indices $i_1$ and $i_m$ are respec-
tively called the left-most position and the right-most of the
occurrence of $S$ in $σ$.

A continuous subsequence $τ$ of a sequence of $σ$ is a token if
no $c$-pair $(a_1, a_2)$ locates outside of $τ$, like $a_1 \cdots τ \cdots a_2$. A
continuous subsequence $T$ of a structure $S$ is a token if no
variable pair $x_j$ and $y_j$ locates outside of $T$. In the sequence
illustrated in Fig. 2, the sets $\{(A, U), (G, C)\}$ and $\{(G, C)\}$
are tokens. The set of tokens in $σ$ ($S$) is denoted by $T(σ)$
(resp. $T(S)$).

3. CONSTRUCTION OF HISTOGRAMS

For a given sequence $σ$ and a structure $S$, we let $N(σ, S)$ be
the number of occurrences of $S$ in $σ$.

In order to calculate $N(σ, S)$ for all $S$, we generate $S$ re-
ursively according to the rules in $G$. For the convenience of
explanation, we introduce a parameter $p$ for representing
$|S|/2$, the number of pairs in $S$, and another $i$ for representing $|T(S)|/2$. We also define two operations $ρ$ and $φ$ for
structures $S$ and $T$ as

$$\begin{align*}
ρ(S) &= xSy, \\
φ(S, T) &= ST.
\end{align*}$$

A brief view of the calculation is as follows: In the calcula-
tion we use a set $S_1$ consisting of pairs of the form $(S, N(σ, σ))$
such that $|S|/2 = p$. The calculation starts with putting
$S_1 = \{(σ, N(σ, σ))\}$, and extend $S_p$ by generating longer
structures.

1. At first consider the case $p = 1$. The only structure
satisfying $|S|/2 = p = 1$ is a variable pair $S = xy$ regarded as
$x_1y_1$, we generate $S_1$ by enumerating all $c$-pair as its
instances.

2. Next consider the cases $p > 1$. Generate instances of
all structures $S$ such that $|S|/2 = p$.

- For the case $i = 1$, update $S_p$ by counting $N(σ, σ)$
  for every structure $S$ with $|T(S)| = i = 1$, which
  are generated by applying $ρ$ to structures $S'$ with
  $|S'|/2 = p - 1$.

- For the case $i = 1$, update $S_p$ by counting $N(σ, S)$
  for every structure $S$ with $|T(S)| = i > 1$, which
  are generated by applying $φ$ to structures $S_1$ and
  $S_2$ with $|S_1|/2 + |S_2|/2 = p$. 

Figure 2: An example of a structure and its instances

\begin{tabular}{ccc}
\hline
&a & b \\
\hline
&\texttt{CAGCACUUGACU} & \texttt{CAGCACUUGACU} \\
\hline
&\texttt{CAGCACUUGACU} & \texttt{CAGCACUUGACU} \\
\hline
\end{tabular}

Figure 3: Examples of tokens

\begin{tabular}{ccc}
\hline
&c & d \\
\hline
&\texttt{CAGCACUUGACU} & \texttt{CAGCACUUGACU} \\
\hline
\end{tabular}
The generation of structures with $|P| = 2$ is illustrated in Fig. 4 and Fig. 5, respectively.

We explain the calculation method more precisely, by changing the form of the tuples corrected in $S_p$. In the calculation we generate tuples of the form

$$(l, r, N(\sigma[l : r], S))$$

for every structure $S$. For example, for the structure and the sequence illustrated in Fig. 2, tuples

$$(1, 2, 2), (2, 12, 1), (2, 11, 1)$$

are generated (Fig. 2).

Let us assume that a sequence $\sigma$ such that $|\sigma| = n$ is given.

At first we set $p = 1$. Since the only structure that we have to treat is $S = xy$, $N(\sigma[xy])$ means the number of c-pairs in $\sigma$ (Algorithm 2). The algorithm is modified so that we can take the least distance of c-pairs into account.

Now we set $p = 2$. The number of structures in this case is obtained by choosing two structures and instances of the case $p = 1$. The methods varies according to the value of $i$. At first we set $i = 1$, and apply the $p$ operator to every structure $S$ generated in the case $p = 1$. We have only one structure $xSy$ (Fig. 4(a)). Next we set $i = 2$, we apply the operator $\phi$. Choose a structure $S$ of the case $p = 1$, we put another structure $T$ with $p = 1$ and $i = 1$ (Fig. 5(a)).

In the case $p = 3$ and $i = 1$, we apply the operator $\rho$ in the same way as in the case $p = 2$ and $i = 1$ every structure with $p = 2$ and $i = 1$.

The method is illustrated in Algorithm 1-4. The algorithm uses three sets $O_S$, $T_p$, $S_p$. The set $O_S$ is used to store tuples $(l, r, N(\sigma[l : r], S))$ for a structure $S$. If $|S|/2 = p$ and $|T(S)| = i$, then $O_S$ is stored in $T_p$. For every $p T_p$ is stored in $S_p$. It is important how to implement the $O_S$, $T_p$, $S_p$, in making the algorithms efficient, and we discuss the ways in the next section.

### 4. ANALYSIS OF THE ALGORITHM

The number of structures consisting of $n$ pairs, that is the $n$-th Catalan number $C_n$, is represented as follows:

$$C_n = a_n + b_n$$

$$a_n = (a_{n-1} + b_{n-1})a_1$$

$$b_n = \sum_{k=1}^{n-1} (a_{n-k} + b_{n-k})a_k$$

$$a_1 = 1, b_1 = 0$$

where $a_n$ is the number of structures consisting of only one token, and $b_n$ is that containing more than one tokens. The equations (3) and (4) respectively correspond to Algorithm 3 and 4. This means that the order of the total calculation is difficult to decrease, and we make our effort to make the calculation for each structure $S$ more efficient.

At first estimate the complexity of enumerating all instances.
of a fixed structure by using a naïve method. Assume that \( |\sigma| = 1 \) for a given sequence. In the case of \( p = 1 \), this means that all pairs in \( \sigma \) and this takes \( O(l^2) \) steps. For \( p > 1 \) every structure is generated by \( \rho \) or \( \phi \). Since structure is stored in the form of the tuple \( 2 \) in \( O_S \), it takes \( O(l^2) \) steps for a fixed argument of \( \rho \) and \( \phi \). Therefore enumerating its instances takes \( O(l^2) \cdot O(l^2) = O(l^4) \) steps.

In order to refine this computation, we take our attention how to keep the tuples in \( \mathcal{O}_S \) and \( \mathcal{O}_S \mathcal{Y} \). The root represents the same left-most position, and leaves are sorted in the ascending order. The trees are also consisting of trees but every tree represents the set of instances takes \( O(l^2) \) steps. For the operator \( \rho \) we can improve in a similar manner. Algorithm 5 and 6.

Figure 7: The method of storing structure and its instances

Algorithm 3: ComposeInNest(int p)

1: for all \( S \) appearing in \( S_p \)
2: \( \mathcal{O}_{S_Y} = \{(t.left, t.right, s.count \cdot t.count) \mid s \in S, t \in \mathcal{O}_S, s < t \} \)
3: (procedure \( \rho \))
4: if \( \mathcal{O}_{S_Y} \neq \emptyset \) then
5: \( S_p \) to \( S_p \)
6: return \( S_p \)

Algorithm 4: ComposeInParallel(int p, int i)

1: for \( k = i - 1 \) to 1 do
2: for all \( S \) appearing in \( \mathcal{T}_{k(i-1)} \), \( T \) appearing in \( \mathcal{T}_{(p-k)} \)
3: \( \mathcal{O}_{ST} = \{(s.left, t.right, s.count \cdot t.count) \mid s \in \mathcal{T}_{k(i-1)}, t \in \mathcal{T}_{(p-k)}, s < t \} \)
4: if \( \mathcal{O}_{ST} \neq \emptyset \) then
5: \( \mathcal{O}_{ST} \) to \( T_{pi} \)
6: return \( T_{pi} \)

Algorithm 5: CompareInNest(Candidate Out, Candidate In, int p, int i, int j)

1: for all \( o.left \in \{\text{Out.lefts}\} \) do
2: for all \( o.right \in \{\text{Out.rights}\} \) do
3: for all \( i.left \in \{\text{In.lefts}\} \) do
4: if \( o.left < i.left \) & \& \( i.left < o.right \) then
5: for all \( r.right \in \{\text{R.rights}\} \) do
6: if \( i.right < r.right \) then
7: hash = \{o.left, o.right, i.count\}
8: \( C_{p+i} = C_{p+i} \cup \text{hash} \)
9: return \( C_{p+i} \)

Left-end position Right-end position # structure instances
1 \( 4 \) 1
2 \( 6 \) 2
3 \( 9 \) 4

Figure 7: The method of storing structure and its instances

5. DATA MINING OF RNA FAMILIES

Here we demonstrate typical data mining tasks, classification and clustering of real RNA families, by using the obtained candidates of RNA secondary structures, and analyze effectiveness of our results. In particular, we focus on the minimum length of distance \( d = |j - i| \) for a pair of \( i\)-th base and \( j\)-th base. We show that dimension reduction is realized by increasing \( d \) experimentally.

5.1 Materials and Methods

All experiments were performed on R version 2.12.1 [10]. We used Mac OS X version 10.6.5 with 2.93 GHz Intel Xeon and 32 GB memory.
Thus we can make data sets in reasonable time. The average and accuracy was obtained by 10 cross-validation. First we package e1071 R the RBF kernel using our constructed data sets. The We performed classification of RNA families by SVM with effect of the difference of length of each sequence. Almost same, hence we can test classification without the sequences of benchmarks from Rfam DR2. For clustering of RNA families, we performed algorithm and DBSCAN is the typical method to find arbitrary shaped clusters. The R packages fpc was used for DBSCAN. We used two RNA families IRE and Histone3. To evaluate results of clustering, we measured the adjusted Rand index (takes values in \([-1, 1]\), to be maximized)\(^7\), which is the typical external criterion, calculated by the R package clues\(^8\). We tuned parameters for DBSCAN and report the best results.

### 5.2 Results and Discussion

We show results of classification in Table 2. We can see that in most of cases, accuracy become higher and higher when \(d\) increases, and the number of attributes decrease monotonically. This means that dimension reduction can be performed effectively with the parameter \(d\) and, moreover, our results of RNA secondary structure candidates can be used effectively for learning classification rules of RNA families. Average length of RNA sequences in CRISPR-DR2 and that in CRISPR-DR3 are almost same, thus we can classify two families with high accuracy even if their average length are similar. Our results are competitive compared to results reported in literatures \([3, 8, 13, 14]\). However, some experimental settings are different, thereby more experiments are needed.

Table 3 shows results of clustering. All adjusted Rand indexes are relatively high, thus this means that our data sets reflect some features of RNA families. Furthermore, the adjusted Rand indexes become higher and higher when \(d\) increases from 1 to 5 in \(K\)-means and 1 to 4 in DBSCAN, hence dimension reduction can be performed effectively.

### 6. CONCLUSION

\(^7\)The newest version 10.0 is available at http://rfam.sanger.ac.uk/

\(^8\)Algorithm 6: CompareInParallel(Candidate \(L\), Candidate \(R\), int \(p\), int \(i\), int \(j\))

```
1: for \(l\).left in \{\(L\).lefts\} do
2: for \(l\).right in \{\(L\).rights\} do
3: for \(r\).left in \{\(R\).lefts\} do
4: if \(l\).right < \(r\).left then
5: for \(r\).right in \{\(R\).rights\} do
6: hash = \{\((l\).left, \(r\).right, \(l\).rightcount - \(l\).left.count)\}
7: \(C_{pij} = C_{pij} \cup hash\)
8: return \(C_{pij}\)
```

\(^9\)Algorithm 7: CompareInNest2(Candidate \(O\), Candidate \(I\), int \(p\), int \(i\), int \(j\))

```
1: for \(o\).left in \{\(O\).lefts\} do
2: \{\(p_{oi} = 0\}\}
3: for \(o\).right in \{\(O\).rights\} do
4: for \(i\).left in \{\(I\).lefts\} do
5: if \(o\).left < \(i\).left then
6: if \(i\).left < \(o\).right then
7: \(i\).right = max(\(p_{oi} = \min(\(I\).rights)\))
8: while next(\(i\).right) < \(o\).right do
9: \(i\).right = next(\(i\).right)
10: if \(p_{oi} \neq \min(\(I\).rights)\) then
11: hash = \{\((o\).left, \(o\).right, \(i\).right.count)\}
12: \(C_{pij} = C_{pij} \cup hash\)
13: \(p_{oi} = i\).right
14: \{\sum right counts by every left index\}
15: return \(C_{pij}\)
```

---

**Figure 8:** New list for improvement

<table>
<thead>
<tr>
<th>Left-end position</th>
<th>Right-end position</th>
<th># structure instances</th>
<th># structure instances</th>
<th>Left-end position</th>
<th>Right-end position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 9:** Comparing structures
Table 1: Benchmark RNA families used for classification and clustering.

<table>
<thead>
<tr>
<th>Family name</th>
<th># RNA sequences</th>
<th>Average length</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRE</td>
<td>247</td>
<td>29.86</td>
</tr>
<tr>
<td>Histone3</td>
<td>381</td>
<td>30</td>
</tr>
<tr>
<td>CRISPR-DR2</td>
<td>64</td>
<td>29.86</td>
</tr>
<tr>
<td>CRISPR-DR3</td>
<td>41</td>
<td>30</td>
</tr>
<tr>
<td>CRISPR-DR4</td>
<td>61</td>
<td>28</td>
</tr>
</tbody>
</table>

We have presented the method of obtaining structure histograms for RNA sequences. The total amount of the time complexity of obtaining the histogram for an RNA sequence of its length 2n is of the n-th Catalan number, but, by designing new data-structures of keeping instances, we show that the value for every structure in the histogram is calculated in the time $O(d^3)$. We also give some experimental results by applying structure histograms obtained from real RNA data to some mining methods and demonstrated the cases that structure histograms work effectively. For applying structure histograms to more practical problems, we have to consider about the dimension reduction more seriously and this is one of our future work.

Enormous RNA sequences are accumulated in previous researches, and many methods have been proposed for predicting RNA secondary structures. However, because of the large size of the RNA data, predicting secondary structures does not catch up analyzing RNA sequences. We expect that our method could contribute semi-automatic prediction of secondary structures and predicting RNA functions.

Acknowledgments

The authors are grateful for valuable discussions and comments to Toshiki Saitoh, Takeaki Uno, and Koichiro Doi. This work is partially supported by Grant-in-Aid for Scientific Research (A) 30230535 from JSPS.

7. REFERENCES

[2] F. Chang, W. Qiu, R. H. Zamar, R. Lazarus, and

Algorithm 8: CompareInParallel2(Candidate L, Candidate R, int p, int i, int j)

1: for l.left in {L.lefts} do
2:   for r.right in {R.rights} do
3:     if l.left < r.right then
4:       crrtIdx = max({R.rights})
5:     for l.right in {L.rights} by desc do
6:       for r.left in {R.lefts} by desc do
7:         if l.right < r.left then
8:           hash = {(l.right, r.left, C. pij)}
9:       crrtIdx = l.right
10:      break
11:    {sum right count by every left index}
12: return C. pij
Figure 11: Scatter plots of a data set from Histone3 and IRE with $d = 6$ (axes are first two attributes). In (a), circle points belong to Histone3, and triangles to IRE, and (b) is a clustering result obtained by $K$-means.

Table 3: Results (adjusted Rand index) of clustering for two families IRE and Histone3.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$-means</td>
<td>0.54</td>
<td>0.54</td>
<td>0.56</td>
<td>0.58</td>
<td>0.62</td>
<td>0.59</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>0.14</td>
<td>0.14</td>
<td>0.75</td>
<td>0.78</td>
<td>0.67</td>
<td>0.64</td>
</tr>
</tbody>
</table>

1994.


