Mining RNA Families with Structure Histograms

Yudai Kawai
Grad. Sch. of Informatics
Kyoto University
Kyoto, 606-8501, Japan
kawai@iip.ist.i.kyoto-u.ac.jp

Mahito Sugiyama
Grad. Sch. of Informatics
Kyoto University
Kyoto, 606-8501, Japan
JSPS Research Fellow
mahito@iip.ist.i.kyoto-u.ac.jp

Akihiro Yamamoto
Grad. Sch. of Informatics
Kyoto University
Kyoto, 606-8501, Japan
akihiro@i.kyoto-u.ac.jp

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 '' .

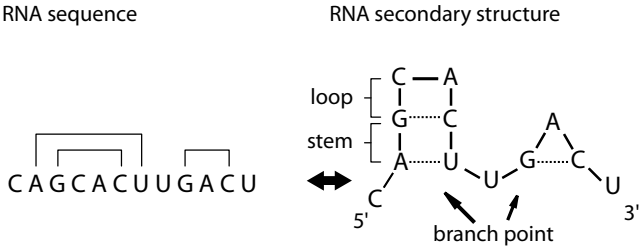

Figure 1: An example of secondary structure of an RNA

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 structures. Because the total time complexity of counting occurrences of all structures would be represented with the Catalan numbers, we propose an efficient of the method for each 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 Σ^+ and call it an *RNA-sequence*. The length n of an RNA-sequence $\sigma = a_1 a_2 \cdots a_n$ is denoted by $n =\mid \sigma \mid$. A number i $(1 \leq i \leq n)$

The 5th International Workshop on Data-Mining and Statistical Science (DMSS2011) Osaka, Japan, March 2011

is called an *occurrence* if it denotes the index of the symbol x_i, and $\sigma[i]$ denotes the symbol a_i, and $\sigma[i:j]$ denotes the continuous subsequence $\sigma = a_i a_{i+1} \cdots a_j$. A pair (i, j) of occurrences of σ such that $i < j$ is called a *candidate pair*, or *c-pair* for short, if $(\sigma[i], \sigma[j])$ 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 complementary 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 $\prec$: $q < p$ if $i_2 < j_2 < i_1 < j_1$, and $q \prec p$ 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:

$$\begin{aligned} R_1 &: S \to xy, \\ R_2 &: S \to xSy, \\ R_3 &: S \to SS \end{aligned}$$

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

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

is called the n-th Catalan number and given as

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

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 $S = xxyyxy$ is regarded as a sequence of variables $x_1 x_2 y_2 y_1 x_3 y_3$.

A *substitution* for a structure $S = z_1 z_2 \cdots z_{2n}$ is a mapping $\theta : \{x_1, \ldots, x_n, y_1, \ldots, y_n\} \longrightarrow \Sigma$ such that $(\theta(x_k), \theta(y_k))$ is a c-pair for every $k = 1, 2, \ldots, n$. A sequence τ in Σ^+ is an *instance* of S if $\tau = S\theta$ for some substitution θ. The structure S *appears* in a sequence $\sigma = a_1 a_2 \cdots a_n$ 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

$$a_{i_1} a_{i_2} \cdots a_{i_m} = S\theta$$

for some substitution θ. The indices i_1 and i_m are respectively 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 $a_1 \cdots \tau \cdots a_2$. 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 $T(\sigma)$ (resp. $T(S)$).

3. CONSTRUCTION OF HISTOGRAMS

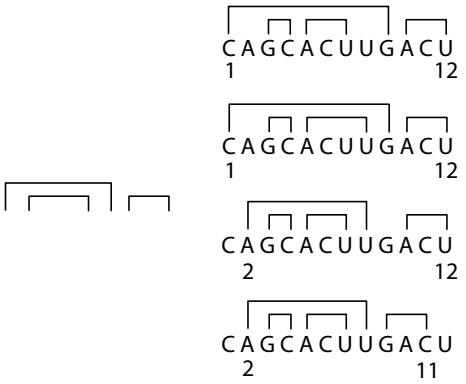

Figure 2: An example of a structure and its instances

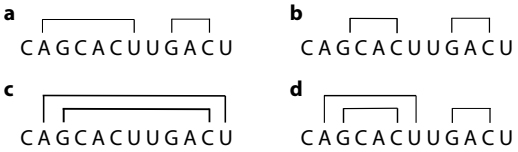

Figure 3: Examples of tokens

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

In order to calculate $N(\sigma, S)$ for all S, we generate S recursively 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

$$\begin{aligned} \rho(S) &= xSy, \\ \varphi(S, T) &= ST. \end{aligned}$$

A brief view of the calculation is as follows: In the calculation we use a set $\mathcal{S}_p$ consisting of pairs of the form $(S, N(\sigma, S)$ such that $|S|/2 = p$. The calculation starts with putting $\mathcal{S}_1 = \{(xy, N(\sigma, xy)\}$, and extend $\mathcal{S}_p$ 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 $x_1 y_1$, we generate $\mathcal{S}_1$ for 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 $\mathcal{S}_p$ by counting $N(\sigma, S)$ 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 $\mathcal{S}_p$ by counting $N(\sigma, S)$ for every structure S with $|T(S)| = i > 1$, which are generated by applying φ to structures S_1 and S_2 with $|S_1|/2 + |S_2|/2 = p$.

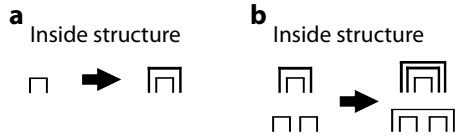

Figure 4: Generating structures S with $|T(S)| = 1$

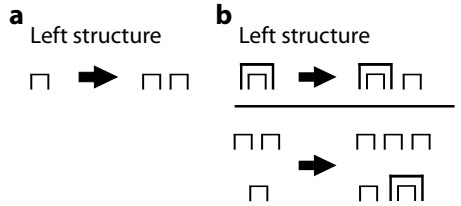

Figure 5: Generating structures S with $|T(S)| > 1$

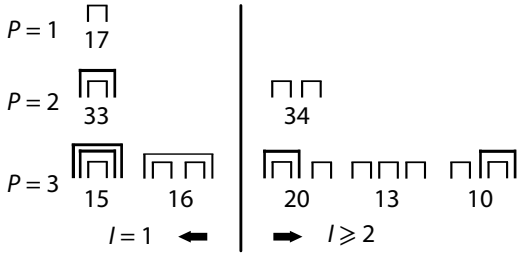

Figure 6: $1 \leq P \leq 3$

The generation of structures with ρ and φ 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 $\mathcal{S}_p$. 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

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

are generated (Fig. 2).

Let us assume that a sequence σ 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 σ (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 ρ 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 φ. 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 ρ in the same way as in the case $p = 2$ and $i = 1$ every structure with

Algorithm 1:
StructureHistogram(a sequence σ, a distance d)

```
1: S_1 = CountPairs(σ)
2: for p = 2 to (|σ| − d)/2 do
3:    T_{p1} = ComposeInNest(p)
4:    S_p = {S_{p1}}
5:    for all i such that 2 ≤ i ≤ p do
6:       T_{pi} = ComposeInParallel(p, i)
7:       add T_{pi} to S_p
8:    end for
9: if S_p = φ then
10:      break
11:   end if
12: end for
```

Algorithm 2: CountPairs(a sequence σ, the least distance d)

```
1: for left = 1 to |σ| − 2 do
2:    for right = |σ| to left + d do
3:       if isPair(σ,left, right) then
4:          add (left, right, 1) to O_{xy}
5:       end if
6:    end for
7: end for
8: T_{11} = {(xy, O_{xy})}
9: S_1 = {T_{11}}
10: return S_1
```

$p = 2$(Fig.4(b)). In the case $p = 3$ and $i \geq 2$, we first apply the φ operator at first to every structure S with $p = 3-1 = 2$ and and every structure T with $p = 1$ and $i = 1$. Then we apply φ to S with $p = 1$ and T with $p = 3 - 1 = 2$ and $i = 1$ (Fig.5(b)). Note that we can choose the structure T with $i = 1$ in the last case. In Fig. 6 we show the structures of the cases $p = 1, 2, 3$ and some examples of numbers of instances.

The method is illustrated in Algorithm 1-4. The algorithm uses three sets $\mathcal{O}_S$, $\mathcal{T}_{pi}$ $\mathcal{S}_p$. The set $\mathcal{O}_S$ 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 $\mathcal{O}_S$ is stored in $\mathcal{T}_{pi}$. For every p $\mathcal{T}_{pi}$ is stored in $\mathcal{S}_p$. It is important how to implement the $\mathcal{O}_S$, $\mathcal{T}_{pi}$ $\mathcal{S}_p$ 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:

$$C_n = a_n + b_n \quad (2)$$
$$a_n = (a_{n-1} + b_{n-1})a_1 \quad (3)$$
$$b_n = \Sigma_{k=1}^{n-1}(a_{n-k} + b_{n-k})a_k \quad (4)$$
$$a_1 = 1, b_1 = 0 \quad (5)$$

where a_n is the number of structures consisting of only one token, and b 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

Algorithm 3: ComposeInNest(int p)

1: **for all** S appearing in $\mathcal{S}_p$
2: $\mathcal{O}_{xSy} = \{(t.left, t.right, s.count \cdot t.count) \mid s \in \mathcal{O}_S, t \in \mathcal{O}_{xy}, s \prec t\}$ (procedure ρ)
3: **if** $\mathcal{O}_{xSy} \neq \emptyset$ **then**
4: add $(xSy, \mathcal{O}_{xSy})$ to $\mathcal{T}_{p1}$
5: add $\mathcal{T}_{p1}$ to $\mathcal{S}_p$
6: **return** $\mathcal{S}_p$

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 $\mathcal{T}_{(p-k)i}$
3: $\mathcal{O}_{ST} = \{(s.left, t.right, s.count \cdot t.count) \mid s \in \mathcal{T}_{k(i-1)}, t \in \mathcal{T}_{(p-k)i}, s < t\}$
4: **if** $\mathcal{O}_{ST} \neq \emptyset$ **then**
5: add $(ST, \mathcal{O}_{ST})$ to $\mathcal{T}_{pi}$
6: **return** $\mathcal{T}_{pi}$

of a fixed structure by using a naïve method. Assume that $|\sigma| = l$ for a given sequence. In the case of $p = 1$, this means that all pairs in σ and this takes $O(l^2)$ steps. For $p > 1$ every structure is generated by ρ or ϕ. Since structure is stored in the form of the tuple 2 in $\mathcal{O}_S$, it takes $O(l^2)$ steps for a fixed argument of ρ and ϕ. Therefore enumerating its instances takes $O(l^2) * O(l^2) = O(l^4)$ steps.

In order to refine these computation, we take our attention how to keep the tuples in $\mathcal{O}_S$. At first we change $\mathcal{O}_S$ as a list $\mathcal{L}_S$ of trees of its depth 1, where every tree represents a subset consisting of tuples which have the same left-most position. The root represents the same left-most position, and leaves are sorted in the ascending order. The trees are also listed in $\mathcal{L}_S$ in the ascending order of their roots (Fig. 7).

This data structure for storing them should make the comparison of two occurrences of structures. Consider the case when we compare S and S'. The left-most position of an instance p is denoted by $p.left$ and its right-most position is $p.right$. With every $q \in S$, we compare every $p \in S'$. At first we fix $q.left$ and $q.right$, and also $p.left$. We check $p.right$ in the ascending order. When all $p.right$ is check, we increment $p.left$, and check $p.right$. After all $p.left$ is checked, we increment $q.right$. We present this method in Algorithm 5 and 6.

Now we improve the time complexity of comparison. The key idea of the improvement is to make another list $\mathcal{M}_S$ for $\mathcal{O}_S$ and use both of $\mathcal{L}_S$ and $\mathcal{M}_S$(Fig. 8). The list $\mathcal{M}_S$ also consisting of trees but every tree represents the set of tuples of the same the right-most position. The construction of every tree in $\mathcal{M}_S$ and the list $\mathcal{M}_S$ is in the symmetric manner of the construction of $\mathcal{L}_S$.

Moreover, each leaf L of a tree in $\mathcal{L}_S$, we give an additional attribute which is the sum of the number of instances above L in the tree.

We change the method as illustrated in Fig. 9. When we construct a new structure S with the ϕ operator from S_1 and S_2, we use $\mathcal{L}_{S_1}$ and $\mathcal{M}_{S_1}$. At first fix the left-most position of S_1 (Fig. 9 (a)) and the right-most position of S_2(Fig. 9 (b)). Then search the left-most position of S_2 in the descending order and simultaneously the right-most position of S_1 (Fig. 9 (c)(c')). In the example illustrated in Fig. 9, we assume that the left-most position of S_1S_2 is 1 and the right-most position of S_1S_2 is 15. Then all the combination of the right-most position of S_1 and the left-most position of S_2 are (9 , 10) , (6 , 7) , (4 , 5). The total number of instances of S is

$$15 = 7 * 1 + 3 * 2 + 1 * 2$$

This method takes $O(2l)$ for every end-pair of S, the total time is $O(l^3)$. For the operator ρ we can improve in a similar manner.

Algorithm 7 and 8 includes the improvement.

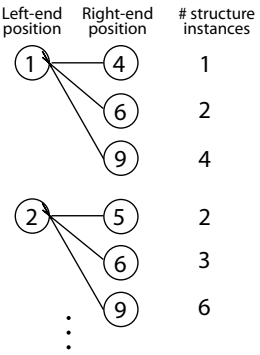

Figure 7: The method of storing structure and its instances

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

1: **for** $o.left$ in $\{Out.lefts\}$ **do**
2: **for** $o.right$ in $\{Out.rights\}$ **do**
3: **for** $i.left$ in $\{In.lefts\}$ **do**
4: **if** $o.left < i.left$ && $i.left < o.right$ **then**
5: **for** $r.right$ in $\{R.rights\}$ **do**
6: **if** $i.right < o.right$ **then**
7: $hash = \{(o.left,\ o.right,\ i.count)\}$
8: $C_{p1j} = C_{p1j} \cup hash$
9: **return** C_{p1j}

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 effectivity 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.

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: $hash = \{(l.left, r.right, l.rightcount \cdot r.left.count)\}$
7: $C_{pij} = C_{pij} \cup hash$
8: **return** C_{pij}

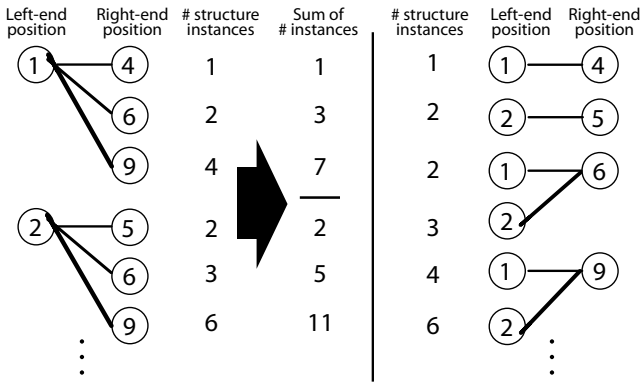

Figure 8: New list for improvement

Every data set was constructed from results obtained from our algorithm (Figure 10). Each row corresponds to each RNA sequence, and each attribute means the number of candidates of secondary structures for the corresponding RNA sequence. RNA families *IRE*, *Histone3*, *CRISPR-DR2*, *CRISPR-DR3*, and *CRISPR-DR4* were collected as benchmarks from Rfam[1] [5] (Table 1), since the number of sequences of *IRE* and *Histone3* are largest in which the average length of RNA sequences is relatively small (under 40). Thus we can make data sets in reasonable time. The average length of sequences in *CRISPR-DR2* and *CRISPR-DR3* are almost same, hence we can test classification without the effect of the difference of length of each sequence.

We performed classification of RNA families by SVM with the RBF kernel using our constructed data sets. The R package `e1071` [9] (the R interface to `libsvm` [1]) was used, and accuracy was obtained by 10 cross-validation. First we classified two families of *IRE* and *Histone3*, second *CRISPR-DR2* and *CRISPR-DR3*, and third three families *IRE*, *Histone3*, and *CRISPR-DR4*.

For clustering of RNA families, we performed K-means and DBSCAN [4], since K-means is the standard clustering 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`

[1]The newest version 10.0 is available at `http://rfam.sanger.ac.uk/`

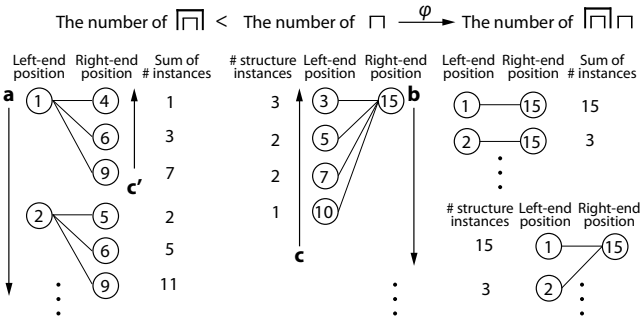

Figure 9: Comparing structures

Algorithm 7: CompareInNest2(Candidate Out, Candidate In, int p, int i, int j)

1: **for** $o.left$ in $\{Out.lefts\}$ **do**
2: $\{p_{i.left}\} = 0$
3: **for** $o.right$ in $\{Out.rights\}$ **do**
4: **for** $i.left$ in $\{In.lefts\}$ **do**
5: **if** $o.left < i.left$ **then**
6: **if** $i.left < o.right$ **then**
7: $i.right = max(p_{i.left}, min(In.rights))$
8: **while** $next(i.right) < o.right$ **do**
9: $i.right = next(i.right)$
10: **if** $p_{i.left} \neq min(In.rights)$ **then**
11: $hash = \{(o.left, o.right, i.right.count)\}$
12: $C_{p1j} = C_{p1j} \cup hash$
13: $p_{i.left} = i.right$
14: {sum right counts by every left index}
15: **return** C_{p1j}

[2]. 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

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

```
for l.left in {L.lefts} do
  for r.right in {R.rights} do
    if l.left < r.right then
      crrtIdx = max({R.lefts})
      for r.left in {R.lefts} by desc do
        for l.right in {L.rights}
          (except more than crrtIdx) by desc do
          if l.right < r.left then
            hash = {(l.left, r.right,
             l.right.count · r.left.count)}
            C_pij = C_pij ∪ hash
            crrtIdx = l.right
            break
  {sum right count by every left index}
return C_pij
```

Table 1: Benchmark RNA families used for classification and clustering.

Family name	# RNA sequences	Average length
IRE	247	29.86
Histone3	381	30
CRISPR-DR2	64	29.86
CRISPR-DR3	41	30
CRISPR-DR4	61	28

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 $O(l^3)$. 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 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

[1] C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines, 2001.

[2] F. Chang, W. Qiu, R. H. Zamar, R. Lazarus, and X. Wang. clues: An R package for nonparametric clustering based on local shrinking. *Journal of Statistical Software*, 33(4):1–16, 2010.

[3] L. Childs, Z. Nikoloski, P. May, and D. Walther. Identification and classification of ncRNA molecules using graph properties. *Nucleic Acids Research*, 37(9):1 – 12, 2009.

[4] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[5] P. P. Gardner, J. Daub, J. G. Tate, E. P. Nawrocki, D. L. Kolbe, S. Lindgreen, A. C. Wilkinson, R. D. Finn, S. Griffiths-Jones, S. R. Eddy, and A. Bateman. Rfam: updates to the RNA families database. *Nucleic Acids Research*, 2008.

[6] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics: A foundation for computer science*, volume 2. Addison-Wesley Reading, MA,

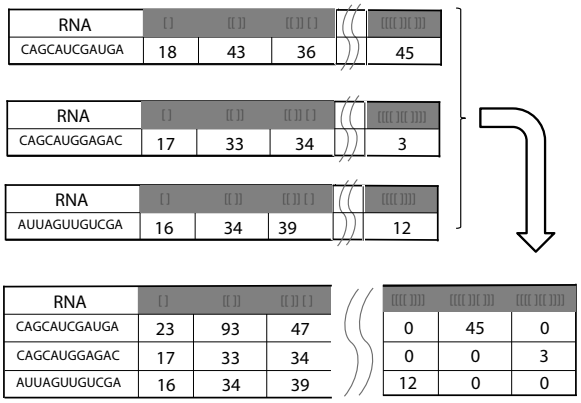

Figure 10: Construction of data sets from the number of candidates of secondary structures for each RNA sequence.

Table 2: Results of classification. The parameter d was increased from 1 to 6, and accuracy of classification was calculated.

	IRE and *Histone3*					
	$d=1$	2	3	4	5	6
accuracy (%)	60.7	96.8	96.8	97.9	99.0	99.7
# attributes	11144	2910	1027	335	170	114
	CRISPR-DR2 and *CRISPR-DR3*					
	$d=1$	2	3	4	5	6
accuracy(%)	61.0	61.0	61.0	92.4	91.4	88.6
# attributes	14101	3851	1053	388	174	94
	IRE, *Histone3*, and *CRISPR-DR4*					
	$d=1$	2	3	4	5	6
accuracy (%)	55.3	55.3	90.3	95.2	98.3	98.8
# attributes	12633	2960	1078	446	177	118

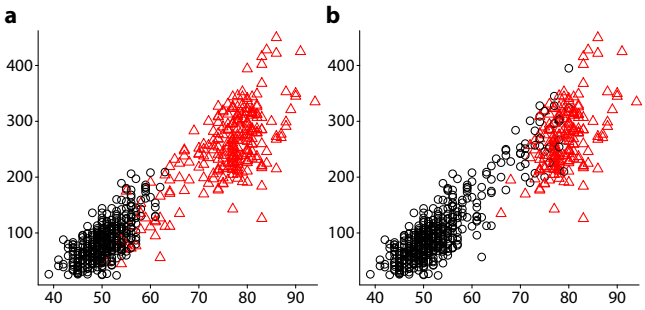

Figure 11: Scatter plots of a data set from *Histone3* and *IRE* with $d = 6$ (axes are first two attributes). In (a), circle points belong to *Histone3*, and triangles to *IRE*, and (b) is a clustering result obtained by K-means.

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

	$d = 1$	2	3	4	5	6
K-means	0.54	0.54	0.56	0.58	0.62	0.59
DBSCAN	0.14	0.14	0.75	0.78	0.67	0.64

1994.

[7] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[8] Y. Karklin, R. F. Meraz, and S. R. Holbrook. Classification of non-coding RNA using graph representations of secondary structure. In *Proceedings of Pacific Symposium on Biocomputing*, volume 10, pages 4 – 15, 2004.

[9] D. Meyer. Support Vector Machines: The interface to libsvm in package e1071, 2010.

[10] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2010.

[11] Y. Sakakibara, M. Brown, R. Hughey, I. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Stochastic context-free grammers for tRNA modeling. *Nucleic Acids Research*, 22(23):5112, 1994.

[12] Y. Tabei, K. Tsuda, T. Kin, and K. Asai. SCARNA: fast and accurate structural alignment of RNA sequences by matching fixed-length stem fragments. *Bioinformatics*, 22(14):1723, 2006.

[13] J. P. Vert. Classification of biological sequences with kernel methods. In Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, and E. Tomita, editors, *Grammatical Inference: Algorithms and Applications*, volume 4201 of *Lecture Notes in Computer Science*, pages 7–18. Springer, 2006.

[14] J. T. Wang and X. Wu. Kernel design for RNA classification using support vector machines. *International Journal of Data Mining and Bioinformatics*, 1(1):57 – 76, 2006.