Efficient Algorithm for Testing Structure Freeness of Finite Set of Biomolecular Sequences

Atsushi Kijima and Satoshi Kobayashi

Graduate School of University of Electro-Communications,
1-5-1, Chofugaoka, Chofu, Tokyo 182-8585, Japan
kijiman@comp.cs.uec.ac.jp, satoshi@cs.uec.ac.jp

Abstract. In this paper we will focus on the structure freeness test problem of finite sets of sequences. The result is an extension of Andronescu's algorithm which can be applied to the sequence design of various DNA computing experiments. We will first give a general algorithm for this problem which runs in $O(n^5)$ time. Then, we will give an evaluation method for sequence design system, which requires $O(n^5)$ time for precomputation, and $O(n^4)$ time and $O(n^5)$ space for each evaluation of sequence sets. The authors believe that this result will give an important progress of efficient sequence design systems.

1 Introduction

Since Adleman's novel biological experiment for solving directed Hamiltonian path problem by DNA molecules was reported([1]), DNA computing paradigm has emerged and progressed while communicating with related fields, such as DNA nanotechnology([19], [15], [7]), biotechnology([5]), etc. One of the most important problems in DNA computing experiments include the design of *structure free* biomolecular sequences which can avoid unwanted secondary structure([6], [8]). In order to develop a sequence design system, we need to devise an efficient algorithm to test the structure freeness of a given set of biomolecular sequences.

Concerning sequence design for DNA computing, there have been many interesting and important works which propose some variants of Hamming distance over biomolecular sequences. And these metrics are used for the evaluation of the sequences([3], [12], [10], etc.). Comparing those Hamming distance approaches, Condon, et al. mathematically formulated a structure freeness test problem of biomolecular sequences at the secondary structure level([8], [2]). This problem is closely related to the prediction problem of RNA secondary structures([9], [11], [16], [20]), and is important in that its efficient algorithms can be applied to the evaluation of sequence sets in sequence design systems.

Andronescu, et al., proposed an $O(m^2n^3)$ time algorithm for testing the structure freeness of a sequence set $S_1 \cdots S_k$, where each S_i is a finite set of sequences of length l_i, $n = \sum_{i=1}^{k} l_i$, and $m = \max\{|S_i| \mid i = 1, ..., k\}$([2]). Kobayashi, et al., gave an $O(m^6n^6)$ time algorithm for testing the structure freeness of a sequence set S^+, where S is a finite set of sequences of length n and $m = |S|$

A. Carbone and N.A. Pierce (Eds.): DNA11, LNCS 3892, pp. 171–180, 2006.

([14]). Furthermore, Kobayashi devised an $O(n^8)$ time algorithm for testing the structure freeness of a regular set of sequences, where n is the number of vertices of graphs for representing the set([13]). (Note that Condon proposed to use a graph for representing a regular set of sequences.)

In spite of this progress in evaluation methods, we still need more efficient algorithms in order to develop an efficient sequence design system. In this paper, we will focus on the structure freeness test problem of finite sets of sequences. The obtained result is an extension of Andronescu's algorithm and can be applied to the sequence design of various DNA computing experiments. We will first give a general algorithm for this problem which runs in $O(n^5)$ time. Then, we will give an evaluation method for sequence design system, which requires $O(n^5)$ time for precomputation, and $O(n^4)$ time and $O(n^5)$ space for each evaluation of sequence sets. The authors believe that this result will give an important progress of efficient sequence design systems.

2 Preliminaries

Σ is an alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ or $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$. A symbol in Σ is called a *base*. A string over Σ represents a DNA or RNA strand with $5' \to 3'$ direction. Consider a string α over Σ. By $|\alpha|$ we denote the length of α. For a finite set X, by $|X|$ we denote the number of elements of X. For an integer i such that $1 \leq i \leq |\alpha|$, by $\alpha[i]$ we denote the ith base of α.

2.1 Secondary Structure

We will partly follow the terminologies and notations used in ([17]). We introduce a relation $\theta \subseteq \Sigma \times \Sigma$ defined by $\theta = \{(\mathtt{A}, \mathtt{T}), \ldots, (\mathtt{T}, \mathtt{G})\}$ for representing Watson-Crick and non-Watson-Crick base pairs of a DNA strand. For the case of an RNA strand, the symbol $\mathtt{T}$ is replaced by $\mathtt{U}$. By (i, j) we denote a hydrogen bond between the ith base and the jth base of a string α. A hydrogen bond is also called a *base pair*. A base pair (i, j) of a string α can be formed only if $(\alpha[i], \alpha[j]) \in \theta$ holds. Without loss of generality, we may assume that $i < j$ for a base pair (i, j). A finite set of base pairs of string α is called a *secondary structure* of α. A string α with its secondary structure T is called a *structured string* and denoted by $\alpha(T)$. For representing the ith base in $\alpha(T)$, we often use the integer i.

In this paper, we consider secondary structures T such that there exist no base pairs $(i, j), (k, l) \in T$ satisfying $i < k < j < l$. In the sequel, we assume that every secondary structure is *pseudo-knot free*.

For a base pair $(i, j) \in \alpha(T)$ and a base r in $\alpha(T)$, we say that (i, j) *surrounds* r if $i < r < j$ holds. For a base pair $(p, q) \in T$, we say that (i, j) surrounds (p, q) if $i < p < q < j$ holds. A base pair (p, q) or an unpaired base r is said to be *accessible* from (i, j), if it is surrounded by (i, j) and is not surrounded by any base pair (k, l) such that (k, l) is surrounded by (i, j). If (p, q) is accessible from (i, j), we write $(p, q) < (i, j)$.

For each base pair $bp = (i, j) \in T$, we define a *cycle* $c(bp)$ as a substructure consisting of the base pair (i, j) together with any base pairs $(p_1, q_1), (p_2, q_2), \ldots, (p_{k-1}, q_{k-1}) \in T$ accessible from (i, j) and any unpaired bases accessible from (i, j). If a cycle $c(bp)$ contains k base pairs including the base pair (i, j), it is said to be k-cycle. In case $k = 1$, we often call it a *hairpin*. In case $k = 2$, it is called *internal loop*. In case $k > 2$, it is called *multiple loop*. In these definitions, the base pair (i, j) is called a *closing base pair* of the cycle. (See Fig. 1).

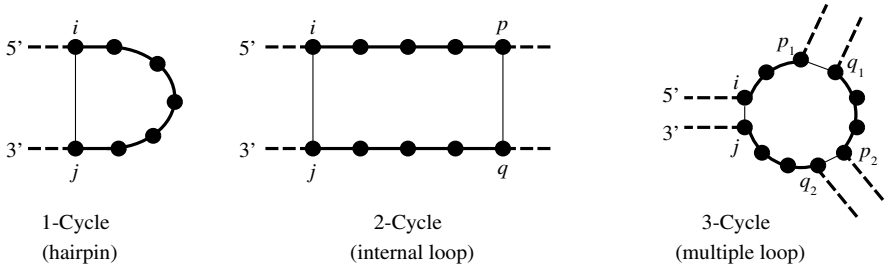

Fig. 1. Secondary Structure

In case of $(1, |\alpha|) \notin T$, the substructure of $\alpha(T)$ consisting of the base pair (i, j) such that (i, j) is not surrounded by any $(p, q) \in T$ $((i, j) \neq (p, q))$ and the unpaired bases such that they are not surrounded by (i, j) is called a *free end structure* of $\alpha(T)$. We do not consider a free end structure because of space constraint.

The *loop length* of a 1-cycle c with a base pair (i, j) is defined as $j - i + 1$. For a 2-cycle c with base pairs $(i, j), (p, q)$ $((p, q) < (i, j))$, we define *loop length* of c as $p - i + j - q + 2$ and define *loop length mismatch* of c as $|(p - i) - (j - q)|$.

By $\overline{\uparrow \alpha \downarrow}$ we denote a 1-cycle consisting of a string α with a base pair between $\alpha[1]$ and $\alpha[|\alpha|]$. By $\overline{\uparrow \alpha \ \beta \downarrow}$ we denote 2-cycle consisting of strings α and β with two base pairs between $\alpha[1]$ and $\beta[|\beta|]$ and between $\alpha[|\alpha|]$ and $\beta[1]$.

3 Free Energy of Secondary Structure

In this paper, we use following simplified functions to assign free energy values to each substructures. We use these simplifications only for the clarity of the algorithm. Experimental evidence is used to determine such free energy values.

1. The free energy $E(c)$ of a 1-cycle c with a base pair (i, j) is dependent on the base pair (i, j) and its loop length l:
$$E(c) = f_1(\alpha[i], \alpha[j]) + g_1(l) \ . \tag{1}$$
2. The free energy $E(c)$ of a 2-cycle c with two base pair $(i, j), (p, q)((p, q) < (i, j))$ is dependent on the base pairs $(i, j), (p, q)$, its loop length l and its loop length mismatch d:
$$E(c) = f_2(\alpha[i], \alpha[j], \alpha[p], \alpha[q]) + g_2(l) + g_3(d) \ . \tag{2}$$

3. The free energy $E(c)$ of a k-cycle c ($k > 2$) with a closing base pair (i, j) and the base pairs $(p_1, q_1), (p_2, q_2), \ldots, (p_{k-1}, q_{k-1})$ accessible from (i, j) is dependent on the base pairs $(i, j), (p_l, q_l)$ $(l = 1, \ldots, k-1)$, the number n_b $(= k)$ of base pairs in c and the number n_u of unpaired bases in c:

$$E(c) = m_1(\alpha[i], \alpha[j]) + \sum_{l=1}^{k} (m_1(\alpha[q_l], \alpha[p_l])) + M_b * n_b + M_u * n_u + C_M \ . \quad (3)$$

In these definitions, the functions $f_1, g_1, f_2, g_2, g_3, m_1$ and the constants M_b, M_u,C_M are experimentally obtained. We assume that M_b, M_u, C_M are non-negative. For each function g_i $(i = 1, 2, 3)$, we assume that g_i is weakly monotonically increasing[1].

We assume that all the above functions are computable in constant time.

Let $c_1, \ldots, c_k$ be the cycles contained in $\alpha(T)$. Then, the free energy $E(\alpha(T))$ of $\alpha(T)$ is given by following:

$$E(\alpha(T)) = \sum_{i=1}^{k} E(c_i) \ . \quad (4)$$

4 Structure Freeness of Finite Regular Set

We will consider the problem of testing whether a given finite regular set of strings is structure free or not. The problem is formally defined in the following way:

Let R be a regular language over Σ. Then, we say that R is *structure free with threshold* D if for any structured string $\alpha(T)$ such that $\alpha \in R$ and T is pseudo-knot free, it holds that $E(\alpha(T)) \geq D$. We have interests in deciding for given R, whether or not R is structure free with threshold D. In Sect. 6, we will give a polynomial time algorithm for solving this problem in the case that R is finite.

For specifying a regular language R, we use a labeled directed graph with initial and final vertices. Let $M = (V, E, \sigma, I, F)$, where V is a finite set of vertices, E is a subset of $V \times V$, σ is a label function from V to Σ, and $I, F \in V$. For $p, q \in V$ and $x \in \Sigma^*$, we write $p \xrightarrow{x} q$ if there is a path with labels x from p to q in M. Note that x contains the labels $\sigma(p)$ and $\sigma(q)$. We write $p \rightarrow q$ if $p \xrightarrow{x} q$ for some $x \in \Sigma^*$. A string α is *accepted* by M if $p \xrightarrow{\alpha} q$ for some $p \in I$ and $q \in F$. This graph representation could be regarded as a Moore type machine with no edge labels. Thus, a set of strings is regular iff it is accepted by a graph M. A graph M is said to be *trimmed* if every vertex is reachable from an initial vertex and has a path to a final vertex.

In this paper, we have interests in testing structure freeness of a finite regular set. Note that a set of strings is finite iff it is accepted by a trimmed and acyclic graph.

[1] This assumption can be extended so that $g_i(l)$ is weakly monotonically increasing within the range $l > L_i$ for some constant L_i. Because of space constraint, we use the simplified assumption.

5 Minimum Free Energy of Substructure

Let R be a finite regular language over Σ and $M = (V, E, \sigma, I, F)$ be a trimmed and acyclic graph accepting R.

We can topologically sort vertices in V in $O(|V| + |E|)$ time. By an integer i, we denote the ith vertex in the topological order. Let $\alpha(T)$ be a structured string such that $\alpha \in R$ and T is pseudo-knot free.

Definition 1. *For $i, j, p, q \in V$, we define:*

(1) $minH(i,j) = \min\left\{E(\overline{\uparrow x \downarrow}) \mid i \xrightarrow{x} j\right\}$,

(2) $minI(i,j,p,q) = \min\left\{E(\underline{\overline{\uparrow x\ y \downarrow}} \mid i \xrightarrow{x} p,\ q \xrightarrow{y} j,\ p \to q\right\}$.

For each $i, j, p, q \in V$ such that there is no $\overline{\uparrow x \downarrow}$ or $\underline{\overline{\uparrow x\ y \downarrow}}$, the value of $minH(i,j)$ or $minI(i,j,p,q)$ is defined as $+\infty$.

For each pair of vertices i, j, we define $Len(i)(j)$ as a set of the length $|x|$ such that $i \xrightarrow{x} j$. For a given graph M, we compute the array Len by the algorithm shown in Fig. 2, where every vertices are sorted in the topological order.

```
Make-Len(M)
begin
   for i,j ∈ V do Len(i)(j) := φ; end
   for (i,j) ∈ E do Len(i)(j) := {2}; end
   for d = 2 to |V| − 1 do
      for i = 1 to |V| − d do
         j := i + d;
         Len(i)(j) := Len(i)(j) ∪ ⋃_{i<k<j, (k,j)∈E} {x + 1 | x ∈ Len(i)(k)};
      end
   end
end
```

Fig. 2. The algorithm `Make-Len`

Since a given graph M is acyclic, for any $i, j \in V$, $|Len(i)(j)| \leq |V|$ holds. So, we can compute an array Len in $O(|V|^2|E|)$ time.

By Definition 1 and by using the array Len, we get the following Proposition 1.

Proposition 1. *For $i, j, p, q \in V$, we define:*

(1) $minH(i,j) = \min\left\{f_1\big(\sigma(i), \sigma(j)\big) + g_1(l) \mid l \in Len(i)(j)\right\}$

(2) $minI(i,j,p,q) = \min\{\ f_2(\sigma(i), \sigma(j), \sigma(p), \sigma(q)) +$
$g_2(l_1 + l_2) + g_3(|l_1 - l_2|) \mid\ l_1 \in Len(i)(p),\ l_2 \in Len(q)(j)\}$.

In case of $Len(i,j) = \phi$ for some $i, j \in V$, we define $minH(i,j) = +\infty$. In case of $Len(i,j) = \phi$ or $Len(p,q) = \phi$ for some $i, j, p, q \in V$, we also define $minI(i,j,p,q) = +\infty$.

Note that the number of elements of a set $\{(x+y, |x-y|) \mid x \in Len(i)(p),\ y \in Len(q)(j)\}$ is $O(|V|^2)$ for $i, j, p, q \in V$.

5.1 Minimum Free Energy of Internal Loop

By Proposition 1, it takes $O(|V|^6)$ time to compute $minI(i,j,p,q)$ for all $i,j,p,q \in V$. We can compute $minI(i,j,p,q)$ more efficiently by computing an array $S_{X,Y}$, $\overline{S}_{X,Y}$ defined as follows:

Definition 2. *Let X and Y be finite sets of positive integers. We define $S_{X,Y}$ as follows:*

$$S_{X,Y} = \big\{(x, \min\{y \in Y \mid x \le y\}) \mid x \in X\big\} \ .$$

Note that we can compute $S_{X,Y}$ in $O(|X| + |Y|)$ time by using the algorithm shown in Fig. 3, and the number of elements of $S_{X,Y}$ is $O(|X| + |Y|)$.

In order to apply $S_{X,Y}$ to computing minimum free energy of strings, we define $\overline{S}_{X,Y}$ as follows:

$$\overline{S}_{X,Y} = \big\{(x,y) \mid (x,y) \in S_{X,Y}\big\} \cup \big\{(x,y) \mid (y,x) \in S_{Y,X}\big\} \ . \tag{5}$$

Theorem 1. *For $i,j,p,q \in V$, we can compute $minI(i,j,p,q)$ in the following way:*

$$minI(i,j,p,q) = \min\{\ f_2(\sigma(i),\sigma(j),\sigma(p),\sigma(q)) + g_2(x+y) + g_3(|x-y|) \mid (x,y) \in \overline{S}_{Len(i)(p),Len(q)(j)}, Len(p)(q) \neq \phi\big\} \ .$$

Proof. Let $X = Len(i)(q)$ and $Y = Len(q)(j)$. It suffices to show that for any $(x,y) \in X \times Y$, there exists $(x',y') \in \overline{S}_{X,Y}$ such that $g_2(x+y) + g_3(|x-y|) \ge g_2(x'+y') + g_3(|x'-y'|)$.

```
Make-S(X, Y)
begin
   S_{X,Y} := φ
   i := |X|;
   j := |Y|;
   y_0 = −∞;
   while i ≥ 1 and j ≥ 1 do
      if x_i ≤ y_j then
         while x_i ≤ y_j and y_{j−1} < x_i and i ≥ 1 do
            S_{X,Y} := S_{X,Y} ∪ (x_i, y_j);
            i := i − 1;
         end
         j := j − 1;
      else
         i := i − 1;
      end
   end
   return S_{X,Y};
end
```

Fig. 3. The algorithm `Make-S`

We consider two cases:

(1) In case of $x \leq y$, let $x' = x$ and $y' = \min\{y'' \in Y \mid x \leq y''\}$. Note that $(x', y') \in S_{X,Y}$. We have $x = x' \leq y' \leq y$. Then, we have $x' + y' \leq x + y$ and $0 \leq y' - x' \leq y - x$. Since functions g_2 and g_3 are weakly monotonically increasing, we have $g_2(x'+y') \leq g_2(x+y)$ and $g_3(y'-x') \leq g_3(y-x)$. Therefore, we can compute minimum free energy $minI(i, j, p, q)$ by using $S_{X,Y}$.
(2) In case of $x > y$, let $x' = \min\{x'' \in X \mid y \leq x''\}$ and $y' = y$. Note that $(y', x') \in S_{Y,X}$. In the same way above, we can also say that we can compute $minI(i, j, p, q)$ by using $S_{Y,X}$.

We can compute $minI$ by using $S_{X,Y}$ or $S_{Y,X}$ in both cases (1) and (2). Therefore, we can compute $minI$ by using $\overline{S}_{X,Y}$. □

Theorem 2. *For each $i, j, p, q \in V$, $minI(i, j, p, q)$ can be computed in $O(|V|)$ time.*

Proof. Since for any $i, j \in |V|$, the number of elements of $Len(i)(j)$ is less than or equal to $|V|$, the number of elements in $S_{Len(i)(p),Len(q)(j)}$ and $S_{Len(q)(j),Len(i)(p)}$ are $O(|V|)$. Therefore, the number of elements of $\overline{S}_{Len(i)(p),Len(q)(j)}$ is $O(|V|)$. □

In real applications of RNA secondary structure prediction([11], [20]), the loop length of internal loops is assumed to be bounded by some constant in order to make the prediction algorithms more efficient. This assumption also enables us to compute $minI(i, j, p, q)$ in constant time for each $i, j, p, q \in V$.

6 Algorithm for Testing Structure Freeness

We will give the algorithm `SFT-FS` for testing the structure freeness of a given finite set of strings represented by a graph $M = (V, E, \sigma, I, F)$. Let an integer i represent the ith element of V in topological order. The algorithm is shown in Fig. 4, where $a(i, j) = m_1(\sigma(i), \sigma(j))$ is the energy contribution of a base pair in a multiple loop.

Our algorithm is based on the dynamic programming approach used in various RNA secondary structure prediction algorithms([11], [20], [17], etc.). While a base adjacent to another base can be determined uniquely for a strand, it does not hold for a set of strands. We consider all possible bases adjacent to a base. Correctness of the algorithm is informally understood as follows:

Let R be a finite set of strings and M be a graph accepting R. Let $\alpha \in R$ be a structured strand $\alpha(T)$ with the minimum free energy $E(\alpha(T))$ such that $i \xrightarrow{\alpha} j$ for some $i \in I, j \in F$. For some $p, q \in V$ such that $p \xrightarrow{\beta} q$, if β is a substring of α, β has the minimum free energy $E(\beta(\hat{T}))$ among all substrands in R such that $p \xrightarrow{\beta} q$, where $\hat{T} \subseteq T$. Otherwise there exist $p \xrightarrow{\beta'} q$ such that $E(\beta'(T')) < E(\beta(\hat{T}))$ and $T' \subseteq T$. Then, we can replace β in α to β' and have $E(\alpha'(T'')) < E(\alpha(T))$, which contradicts the minimality of the free energy $E(\alpha(T))$. The algorithm computes such minimum free energy from smaller to

```
Init(M)
begin
   Topological-Sort(M) ;
   compute Len(i)(j) for all i, j ∈ V by calling Make-Len(M) ;
   compute S_{Len(i)(p),Len(q)(j)}
    by calling Make-S(Len(i)(p), Len(q)(j))  for all i, j, p, q ∈ V;
   compute minH(i, j), minI(i, j, p, q) for all i, j, p, q ∈ V;
end

SFT-FS(M)
begin
   Init(M);
   for d = 1 to |V| − 1 do
      for i = 1 to |V| − d do
         j = i + d;
         (I) C[i, j] = min { minH(i, j) ,
                           { min_{i<p<q<j} {minI(i, j, p, q) + C[p, q]} ,
                           { min_{i<i'<j'<j, (i,i'),(j',j)∈E} {FM[i', j'] + a(i, j)}.
         (II) F[i, j] = min { C[i, j] ,
                            { min_{i<k<k'<j, (k,k')∈E} {FM[i, k] + FM[k', j]} .
         (III) FM[i, j] = min { M_b + C[i, j] ,
                              { min_{i<i'<j, (i,i')∈E} {M_c + FM[i', j]} ,
                              { min_{i<j'<j, (j',j)∈E} {M_c + FM[i, j']} ,
                              { min_{i<k<k'<j, (k,k')∈E} {FM[i, k] + FM[k', j]} .
      end
   end
   if there exist F[i, j] < threshold D for some i ∈ I, j ∈ F return 'No';
   else return 'Yes';
end
```

Fig. 4. The algorithm SFT-FS

larger substructures with the recurrences (I)–(III) applied to topologically sorted vertices.

We can run Init(M) in $O(|V|^5)$ time, SFT-FS(M) in $O(|V|^4)$ time, and it costs $O(|V|^5)$ time in total. By using the constant upper bound assumption on loop length in Sect. 5.1, we can run Init(M) in $O(|V|^4)$ time.

7 Application to Strand Design

Our algorithm requires more time in Init(M) than SFT-FS(M). Once the initialization Init(M) is done, we can evaluate strands more efficiently. Even if we change a label function σ for a vertex, it is not necessary to compute $\overline{S}_{Len(i)(p),Len(q)(j)}$ again. Furthermore, we can compute $minI(i, j, p, q)$ for all possibilities of label function σ which has four possibilities $\sigma(i) =$ A, $\sigma(i) =$ C,

$\sigma(i) = \mathtt{G}$ or $\sigma(i) = \mathtt{T}$ for a vertex i. Then, time to compute $minI$ is $O(|V|^5)$. These observations lead us to a method for strand design shown in Fig. 5. In this search algorithm, we can evaluate a set of strings R.

```
Strand-Design(M)
begin
   Init(M) with all possibilities of label function;

   while SFT-FS(M) returns 'No' do
      change a label of a randomly selected vertex;
   end
   return M;
end
```

Fig. 5. Random strand design algorithm

In this `Strand-Design`(M), a random search is used for finding a structure free set of sequences. In real applications to sequence design, we should use more sophisticated search strategies, such as stochastic local search([18]), genetic algorithm([4]), etc.

8 Conclusion

We give an efficient algorithm for testing the structure freeness of a finite set of strands. We also give a method for strand design generating a finite set of structure free strands. Our future works will include the improvement of the algorithm for computing $minI$ and the implementation of the strand design system based on the results presented in this paper.

References

1. L. Adleman, Molecular Computation of Solutions to Combinatorial Problems. *Science*, vol.266, pp.1021-1024, 1994.
2. M. Andronescu, D. Dees, L. Slaybaugh, Y. Zhao, A. Condon, B. Cohen, and S. Skiena, Algorithms for Testing That Sets of DNA Words Concatenate without Secondary Structure. *Proc. of The 9th International Meeting on DNA Based Computers*, LNCS, vol.2568, pp.182-195, 2003.
3. M. Arita and S. Kobayashi, DNA sequence design using templates. *New Generation Computing*, vol.20, pp.263-277, 2002.
4. M. Arita, A. Nishikawa, M. Hagiya, K. Komiya, H. Gouzu, and K. Sakamoto, Improving sequence design for DNA computing. *Proc. of Genetic and Evolutionary Computation Conference 2000*, pp.875-882, 2000.
5. Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, E. Shapiro, An autonomous molecular computer for logical control of gene expression. *Nature*, vol.429, pp.423-429, 2004.
6. A. Brenneman and A. E. Condon, Strand Design for Bio-Molecular Computation. *Theoretical Computer Science*, vol.287, pp.39-58, 2002.

7. A. Carbone and N. C. Seeman, Circuits and programmable self-assembling DNA structures. *Proc. Natl. Acad. Sci. USA*, vol.99, pp.12577-12582, 2002.
8. A. E. Condon, Problems on RNA Secondary Structure Prediction and Design. *Proc. of ICALP'2003*, Lecture Notes in Computer Science, vol.2719, pp.22-32, 2003.
9. R. M. Dirks, N. A. Pierce, An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *Journal of Computational Chemistry*, vol.25, pp.1295-1304, 2004.
10. A. G. D'yachkov, A. J. Macula, W. K. Pogozelski, T. E. Renz, V. V. Rykov, D. C. Torney, A weighted insertion-deletion stacked pair thermodynamic metric for DNA codes. *Preliminary Proc. of Tenth International Meeting on DNA Computing*, pp.142-151, 2004.
11. I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster, Fast Folding and Comparison of RNA Secondary Structures (The Vienna RNA Package). *Monatshefte für Chemie*, vol.125, pp.167-188, 1994.
12. L. Kari, S. Konstantinidis, and P. Sosík, Bond-free languages: formalizations, maximality and construction methods. *Preliminary Proc. of Tenth International Meeting on DNA Computing*, pp.16-25, 2004.
13. S. Kobayashi, Testing structure freeness of regular sets of biomolecular sequence. *Preliminary Proc. of Tenth International Meeting on DNA Computing*, pp.395-404, 2004.
14. S. Kobayashi, T. Yokomori, and Y. Sakakibara, An Algorithm for Testing Structure Freeness of Biomolecular Sequences. *Aspects of Molecular Computing — Essays dedicated to Tom Head on the occasion of his 70th birthday*, Springer-Verlag, LNCS, vol.2950, pp.266-277, 2004.
15. C. Mao, T. H. LaBean, J. H. Reif, and N. C. Seeman, Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, vol.407, pp.493-496, 2000.
16. J. S. McCaskill, The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, vol.29, pp.1105-1119, 1990.
17. D. Sankoff, J. B. Kruskal, S. Mainville, and R. J. Cedergen, Fast Algorithms to Determine RNA Secondary Structures Containing Multiple Loops. *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Editors, Chapter 3, pp.93-120, 1983.
18. D. C. Tulpan, H. H. Hoos, and A. E. Condon, Stochastic local search algorithms for DNA word design. *Proc. 8th International Workshop on DNA-Based Computers*, LNCS 2568, pp.229-241, 2002.
19. E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman, Design self-assembly of two-dimensional DNA crystals. *Nature*, vol.394, pp.539-544, 1998.
20. M. Zuker, On finding all suboptimal foldings of an RNA molecule. *Science*, vol.244, pp.48-52, 1989.