Chaotic Motif Sampler: Discovering Motifs from Biological Sequences by Using Chaotic Neurodynamics

Takafumi MATSUURA and Tohru IKEGUCHI

Graduate School of Science and Engineering, Saitama University
225 Shimo-ohkubo, Sakura-ku, Saitama-city 338-8570, JAPAN
Email: takafumi@nls.ics.saitama-u.ac.jp, tohru@ics.saitama-u.ac.jp

Abstract—To determine essential biological information in genomic sequences, local multiple alignment is often solved in bioinformatics. However, it has been proved that the local multiple alignment is an $\mathcal{NP}$-hard problem. Then, heuristic algorithms are proposed to solve the alignment problem within a reasonable time frame. To find near optimal solutions for $\mathcal{NP}$-hard combinatorial optimal problems such as solving traveling salesman problems and quadratic assignment problems, chaotic search methods have been proposed and these methods show good performance. By using the concept of the chaotic search, we have already proposed a motif extraction method called "Chaotic Motif Sampler." In this paper, to improve the performance of the CMS, we propose a different type of motif extraction method by using chaotic dynamics.

1. Introduction

The Human Genome Project has completed in April 2003. The aim of this project is to identify all of approximately 20,000-25,000 genes and to determine the sequences of the 3 billion bases in human DNA. It is now desired to determine which parts of DNA sequences contain biologically important information. One of the popular techniques for deciding such parts is to find a conserved pattern, which is called a "motif." Along the evolution process, DNA sequences have been undergone mutations, insertions, losses, and substitutions by various causes. However, the motifs have been preserved even beyond the species. If the motifs are lost in the evolution process, the resulting creatures may become extinct because the essential proteins for survival cannot be created. Thus, functional roles of genes are clarified by discovering motifs from biological sequences.

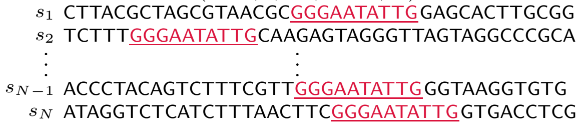
The motif extraction problem (MEP) is mathematically described as follows: we have a biological data set $S = \{s_1, s_2, \cdots, s_N\}$, where s_i is the ith sequence. Each sequence consists of m_i $(i = 1, 2, \cdots, N)$ elements, and the length of the motif is L (Fig. 1). We extract a motif from each sequence, then a similarity of the extracted motifs is calculated by the following objective function:

$$E = \frac{1}{L}\sum_{k=1}^{L}\sum_{\omega\in\Omega} f_k(\omega)\log_2\frac{f_k(\omega)}{p(\omega)}, \qquad (1)$$

where $f_k(\omega)$ is the number of appearances of an element $\omega \in \Omega$ at the kth position of motifs, Ω is a set of 4 bases in case of DNA sequence or a set of 20 amino acids for a protein sequence, $p(\omega)$ is the background probability of appearance of the element ω. If the extracted motifs are similar, the objective function takes a large value because the values of $f_k(\omega)$ become large.

One of the most simple methods for finding motifs is an enumeration method. However, if the length of the motif increases, it is almost impossible to find the motifs by the enumeration method because the number of motif patterns exponentially increases. In case of DNA or RNA sequence, the number of possible motif patterns is 4^L. For a protein sequence, the number becomes 20^L. It is proved that the MEP is an $\mathcal{NP}$-hard problem [1]. Thus, many methods have been developed to find the motifs, for example, the Gibbs Site Sampler [12], the MEME [13], the CONSENSUS [14], and the Neighborhood Optimization for Multiple Alignment Discovery [11].

As for heuristic algorithms for solving combinatorial optimization problems, many methods have been proposed and show good results, for example, a tabu search [2, 3], an exponential tabu search [7], and a chaotic search [5, 7]. In earlier studies, it has already been shown that the chaotic search method can obtain good near optimal solutions for $\mathcal{NP}$-hard combinatorial optimization problems such as the traveling salesman problem [5, 6] and the quadratic assignment problem [7]. Then, we have already proposed a motif extraction method called "Chaotic Motif Sampler (CMS)" by using a concept of the chaotic search [8–10]. In this paper, to improve the performance of the CMS, we propose a different type of motif extraction method by using chaotic dynamics.



```
         ←———— m_i (i = 1, 2, ···, N − 1, N) ————→
s_1      CTTACGCTAGCGTAACGCGGGAATATTGGAGCACTTGCGG
s_2      TCTTTGGGAATATTGCAAGAGTAGGGTTAGTAGGCCCGCA
 ⋮                         ⋮
s_{N−1}  ACCCTACAGTCTTTCGTTGGGAATATTGGGTAAGGTGTG
s_N      ATAGGTCTCATCTTTAACTTCGGGAATATTGGTGACCTCG
```

Figure 1: An example of the motif extraction problem. A, C, G, and T denote Adenine, Cytosine, Guanine, and Thymine, respectively. In this example, the motif is "GGGAATATTG".

2. Chaotic Motif Sampler

We have already proposed a motif extraction method called "Chaotic Motif Sampler" by using chaotic dynamics [8–10]. In this paper, we propose a new chaotic search method to find motifs. In both methods, to realize a chaotic search, a chaotic neural network composed of a chaotic neuron model [4] is constructed. The chaotic neuron exhibit a refractory effect, which is one of the important characteristics of real biological neurons: once a neuron fires, this neuron becomes hard to fire for a while. The chaotic neurons are assigned to the head positions of all motif candidates (Fig.2). Thus, the chaotic neural network is constructed by the number of $\sum_{i=1}^{N}(m_i - L + 1)$ chaotic neurons (Fig.2).

An output state of the chaotic neuron is defined as follows:

$$x_{ij}(t+1) \quad = \quad f(y_{ij}(t+1)), \qquad (2)$$

where $x_{ij}(t)$ is the output state of the (i, j)th chaotic neuron at time $t+1$, $f(y) = 1/(1+\exp(-y/\epsilon))$ is a sigmoidal function, and $y_{ij}(t+1) = \xi_{ij}(t+1) + \zeta_{ij}(t+1)$ is an internal state of the (i, j)th chaotic neuron at time $t+1$. If $x_{ij}(t+1) > \frac{1}{2}$, the (i, j)th neuron fires and the corresponding motif candidate is selected as a motif. On the other hand, if the (i, j)th neuron is resting ($x_{ij}(t+1) \leq \frac{1}{2}$), the (i, j)th position of motif candidate is not selected. The $\xi_{ij}(t+1)$ is a gain effect of the (i, j)th neuron at time $t+1$. The $\zeta_{ij}(t)$ is a refractory effect of the (i, j)th neuron at time $t+1$. These effects have different effects for firing of a chaotic neuron.

The gain effect is defined as follows:

$$\xi_{ij}(t+1) \quad = \quad \beta\Big(E_{ij}(t) - \hat{E}\Big), \qquad (3)$$

where β (> 0) is a scaling parameter of the gain effect; $E_{ij}(t)$ is an objective function of the motif extraction problem (Eq.(1)), when a motif candidate position is moved to the jth position in the sequence s_i; and $\hat{E}$ is a current value of the objective function (Eq.(1)). If the jth motif candidate in the ith sequence is better than the current motif candidate, this function becomes positive; a positive value leads to firing of the neuron. Thus, if we use only the gain effect, the algorithm becomes greedy.

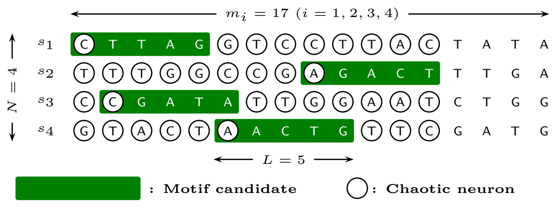

$m_i = 17$ $(i = 1, 2, 3, 4)$
s_1 C T T A G G T C C T T A C T A T A
s_2 T T T G G C C G A G A C T T T G A
s_3 C C G A T A T T G G A A T C T G G
s_4 G T A C T A A C T G T T C G A T G
$N = 4$
$L = 5$
: Motif candidate
: Chaotic neuron

Figure 2: A coding scheme of assigning motif positions to chaotic neurons. In this example, the number of sequences is $N = 4$; the length of each sequence is $m_i = 17$; and the motif length is $L = 5$. Then, the number of chaotic neurons is $4 \times (17 - 5 + 1) = 52$.

However, we do not find good solution only by the gain effect due to the local minimum problem. To escape from the local minima, each chaotic neuron is assigned to a refractory effect. The refractory effect is expressed as follows:

$$\zeta_{ij}(t+1) \quad = \quad -\alpha \sum_{d=0}^{t} k_r^d x_{ij}(t-d) + \theta \qquad (4)$$

where α is a positive parameter, k_r is a decay parameter ($0 < k_r < 1$), and θ is a threshold value. The refractory effect is one of the important properties of real biological neurons: once a neuron fires, a certain period of time must pass before the neuron can fire again. In Eq.(4), if the neuron has fired many times in the past, $\sum_{d=0}^{t} k_r^d x_{ij}(t-d)$ takes positive value. As a result, the first term on the right-hand side of Eq.(4) becomes a negative value. This effect causes the neuron to enter a relatively resting state.

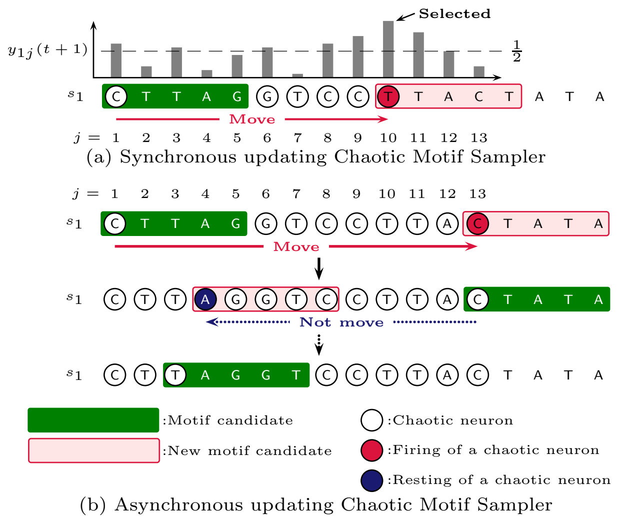
We introduced another operation called phase shift (PS). PS is commonly used for avoiding sub-optimal solutions [11, 12]. In this operation, all motif candidates are shifted a few positions to the left or to the right simultaneously.

The algorithm of the CMS [8–10] can be described as follows:

1. For a given set of sequences $S = \{s_1, s_2, ..., s_N\}$, let the number of sequences be denoted by N; the length of each sequence, by m_i ($i = 1, 2, \cdots, N$); and the length of the motif, by L.
2. The position of an initial motif candidate is randomly selected in each sequence.
3. To shift the position of the motif candidate in each sequence, the sequence s_i is selected cyclically.
4. To change the motif position in sequence s_i, the internal state of all the chaotic neurons in the sequence s_i is calculated. Then, the jth neuron whose internal state is maximum is selected in the sequence s_i. If the (i, j)th neuron fires, the corresponding motif candidate becomes a new motif position and the value of $\hat{E}$ is updated. If $\hat{E}$ is the best solution, PS is applied to the solution. Figure 3(a) shows a procedure of how to change a motif position in the sequence s_1.
5. Repeat steps 3-4 for all sequences.
6. PS is applied to a current solution.
7. The procedure of a single iteration in the CMS is finished. Repeat steps 3-7 until the number of iterations is satisfied.

At step 4 in the above algorithm, the internal state of all the chaotic neurons in the same sequence is updated at the same time. Thus, we call this algorithm *Synchronous updating Chaotic Motif Sampler* (SCMS).

In the SCMS, the maximum internal state of the chaotic neuron is selected as a new motif candidate, even though other neurons fire. As a result, although such neurons are not selected as a new motif candidate, these neurons are prohibited to fire at next time because of the refractory effect. Thus, to improve this problem, we propose a new motif extraction method. The difference between the SCMS


Selected
$y_{1j}(t+1)$
$\frac{1}{2}$
s_1
Move
$j = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13$
(a) Synchronous updating Chaotic Motif Sampler
Not move
:Motif candidate
:New motif candidate
:Chaotic neuron
:Firing of a chaotic neuron
:Resting of a chaotic neuron

(b) Asynchronous updating Chaotic Motif Sampler

Figure 3: The graphical representations of (a) the SCMS and that of (b) the ACMS. In (a), at first, the internal state of all the neurons in the sequence s_1 is calculated. Then, the $(1, 10)$th position becomes a new motif position because the internal state of the $(1, 10)$th neuron is maximum and it fires. In (b), at first, the $(1, 10)$th neuron is selected randomly. Then, the $(1, 10)$th position becomes a new motif candidate position because the state of the $(1, 10)$th neuron is firing. Next, a neuron is selected from the sequence s_1 again. In this case, the $(1, 4)$th neuron is selected. However, this position cannot be selected because its state is resting. Until the internal state of all neurons in the sequence s_1 is updated, a neuron is selected randomly.

and the new method is only in step 4 of the algorithm of the SCMS. We call the new method *Asynchronous updating Chaotic Motif Sampler* (ACMS) because the internal state of the chaotic neurons in the same sequence is not be updated at the same time. Then, step 4 in the algorithm of the ACMS is described as follows:

4. To shift the position of the motif candidate in each sequence, the sequence s_i is selected cyclically.
   (a) The jth neuron is randomly selected from the sequence s_i. If the (i, j)th neuron fires, the corresponding motif candidate becomes a new motif position, and the value of $\hat{E}$ is updated. If $\hat{E}$ is the best solution, PS is applied to the solution. Figure 3(b) shows a procedure of how to change a motif position in the sequence s_1.
   (b) Repeat step 4(a) for all neurons in the sequence s_i.

The other steps are same as the algorithm of the SCMS.

2.1. Simulations and Results

To investigate the performance of the SCMS and that of the ACMS, we prepared real protein sequences [12]. The data set has 30 sequences composed of several number of amino acids. The length of the motifs is 18. If the motifs are extracted exactly, the objective function (Eq.(1)) takes 1.65.

In this simulation, the values of parameters α, k_r, β, and ϵ are set to several values. The other parameter θ is fixed to 1. The width of PS is set to 6. Each motif is changed 500 times in one trial. We conducted simulations using the gcc compiler on a Mac Pro (2×2.8 GHz Quad-Core Intel Xeon) with 2GB memory running Mac OS X 10.5.7.

Figure 4 shows results of the SCMS without PS and that of the ACMS without PS. From Fig. 4, the motifs are exactly extracted, when we appropriately set to the values of parameters. However, the effective parameters sets are different from each method. We cannot find motifs for $\beta = 80$. The reason is that the SCMS and the ACMS cannot escape from local minima because the strength of the greedy effect is stronger than that of the refractory effect. In other word, searching approach is similar to steepest descent method in case of $\beta = 80$.

Figure 5 shows results of the SCMS with PS and that of the ACMS with PS. From Figs. 4 and 5, the performances of both methods become better, especially in case that $\beta = 80$ and $\epsilon = 0.001$. The ACMS can discover motifs for almost all parameters sets. From these results, PS is effective for the SCMS and the ACMS. However, from Figs. 4 and 5, if the success rates of the SCMS without PS and that of the ACMS without PS take almost 0%, the success rates of these methods with PS are also low. Thus, to search solution space effectively, it is necessary to control the strength of the gain effect and that of the refractory effect.

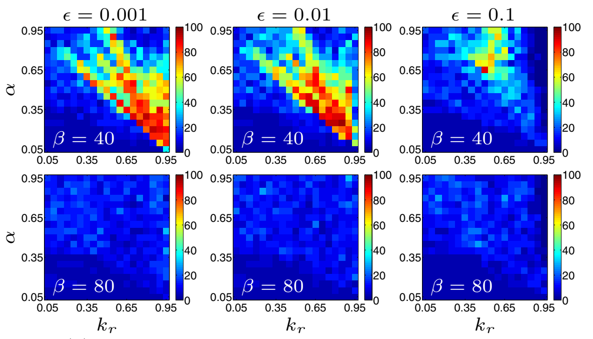
Figure 6 shows CPU-times of the SCMS with PS and that of the ACMS with PS, when the success rates of each method are 100%. From Figs. 5 and 6, CPU times of the SCMS are shorter than that of the ACMS. However, as for a region in which the success rate is 100%, the region of the ACMS is larger than that of the SCMS.

3. Conclusion

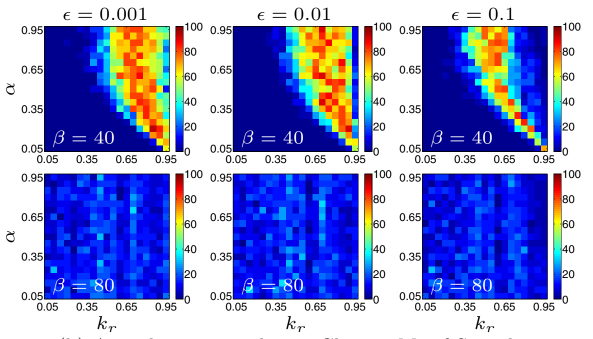
In this paper, we proposed a new method by using chaotic dynamics for solving motif extraction problems. which is one of the important issues in genome information science. The results indicate that the proposed method can find motifs very high rate 100%, when we have to set optimal parameters. However, the good parameters sets is relatively large.

Acknowledgments

The research of T.M. is partially supported by Grant-in-Aid from the JSPS Fellows (No.20·6863). The research of T.I. is partially supported by Grant-in-Aid for Scientific Research (B) (No.20300085) from the JSPS.

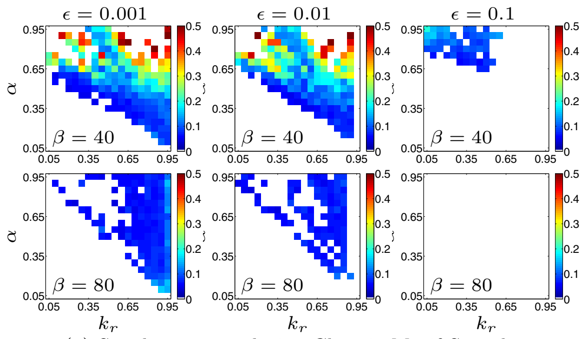


(a) Synchronous updating Chaotic Motif Sampler

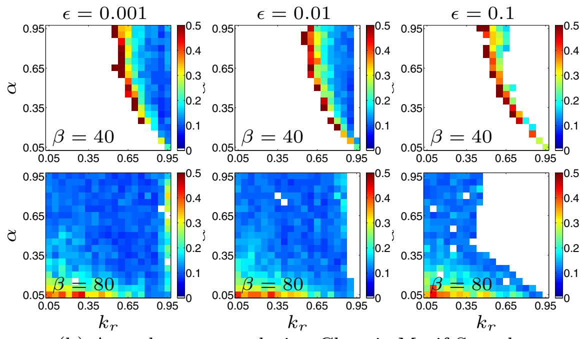


(b) Asynchronous updating Chaotic Motif Sampler

Figure 4: Performance of the SCMS without PS and that of the ACMS without PS. The average probabilities (%) of finding motifs in 30 trials are indicated by color bar.

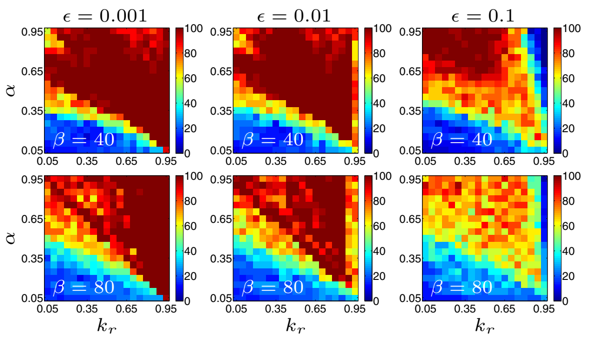


(a) Synchronous updating Chaotic Motif Sampler

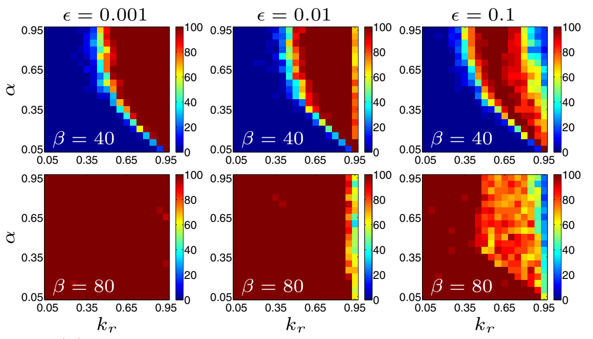


(b) Asynchronous updating Chaotic Motif Sampler

Figure 6: Average CPU times (sec) of the SCMS with PS and that of the ACMS with PS. Although the average CPU times are indicated by color bar, if the average probabilities of finding motifs is not 100%, corresponding regions are drawn in white.



(a) Synchronous updating Chaotic Motif Sampler



(b) Asynchronous updating Chaotic Motif Sampler

Figure 5: Performance of the SCMS with PS and that of the ACMS with PS. The average probabilities (%) of finding motifs in 30 trials are indicated by color bar.

References

[1] Akutsu, T. (2000): *Proc. of the 4th Annual Int. Conf. Molecular Biology*, 1–7.

[2] Glover, F. (1989): *ORSA J. on Comp.*, **1**, 190–206.

[3] Glover, F. (1989): *ORSA J. on Comp.*, **2**, 4–32.

[4] Aihara, K. et al. (1990): *Phys. Lett. A*, **144**, 333–340.

[5] Hasegawa, M. et al. (1997): *PRL*, **79** (12), 2344–2347.

[6] Hasegawa, M. et al. (2002): *Neural Networks*, **15** (2), 271–283.

[7] Hasegawa, M. et al. (2002): *EJOR*, **39** (3), 543–556.

[8] Matsuura, T. et al. (2005): *Proc. of MIC2005*, 677–682.

[9] Matsuura, T. and Ikeguchi, T. (2006): *LNCS*, **4224**, 1103–1110.

[10] Matsuura, T. et al. (2007): *Proc. of MIC2007*.

[11] Hernandez, D. et al. (2005): *EJOR*, **185**, 1276–1284.

[12] Lawrence, C.E. et al. (1993): *Science*. **262**, 208–214.

[13] Bailey, T.L. and Elkan, C. (1995): *Machine Learn*, **21**(1/2), 51–8.

[14] Hertz, G.Z. and Stormo, G.D. (1999): *Bioinformatics*, **15**, 563–577.