Virtual DNA Simulator and Protocol Design by GA

Akio Nishikawa*
Dept. of Information Science,
Graduate School of Science,
University of Tokyo,
Tokyo, JAPAN

Masami Hagiya
Dept. of Information Science,
Graduate School of Science,
University of Tokyo,
Tokyo, JAPAN

Masayuki Yamamura
Dept. of Computational
Intelligence and Systems Science,
Interdisciplinary Graduate School
of Science and Engineering,
Tokyo Institute of Technology,
Yokohama, JAPAN

Abstract

Many algorithms and protocols for DNA computing have been proposed so far, but most of them remain mere proposals and their feasibility has not yet been verified. Even in cases when *in vitro* experiments are possible, it is desirable to verify the feasibility in advance. We developed a simulator to aid those who design algorithms and protocols for DNA computing. In this simulator, abstract sequences instead of real DNA sequences are used to represent molecules in order to increase efficiency of simulations. It consists of two main parts, one for finding reactions among existing molecules and generating new ones, and the other for numerically solving differential equations to calculate the concentration of each molecule. The two parts rely on each other. In particular, the former avoids a combinatorial explosion by setting a threshold on concentrations of molecules that can take part in reactions. Some simulation results are also presented: computation of Boolean circuits, formation of DNA tiles and simulation of polymerase chain reaction (PCR). As for PCR, we also tried to find good protocols for PCR amplification using Genetic Algorithm (GA).

1 Introduction

Many algorithms and protocols for DNA computing have been proposed so far. Some have been actually implemented in *in vitro* experiments, but most remain mere proposals and their feasibility has not yet been verified. Even when *in vitro* experiments are possible, it is desirable to verify the feasibility of algorithms and protocols in advance. In particular, as the problem size of DNA computing grows larger, it becomes more difficult to avoid unexpected mis-hybridization or other types of unexpected reactions. Since it is almost impossible to check all possible reactions by hand, tools for predicting such reactions are needed.

In the field of DNA computing, some simulation systems have already been developed. For example, Winfree implemented a simulator which finds the proper experimental conditions for efficient assembly of DNA tiles [8]. The simulator is based on a stochastic model, and helps design DNA tiles by providing suggestions on whether they can assemble as expected.

In this study, we aim to develop a general-purpose simulator for DNA computing. Unlike Winfree's simulator, which is specific to DNA tiles, ours is intended to cover a large class of algorithms and protocols consisting of major reactions for DNA computing, i.e., hybridization, denaturation, restriction, ligation and extension. A similar effort is being paid by Zhang and Shin [10], but functions of their system are not reported in detail.

There are many biological simulation systems that solve differential equations numerically. For example, BCT is famous as a simulator for bacterial chemotaxis [1]. Recently, systems that simulate all the known reactions in a cell have been developed to predict cell behaviors. E-cell is one of such systems [6]. However, these systems require that all the molecules be known in advance and do not generate new molecules during a simulation. In DNA computing, however, new molecules are often produced as computation proceeds. Therefore, simulation systems for DNA computing should be able to generate new molecules by enumerating possible reactions between

* Dept. Information Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, JAPAN. (e-mail: nisikawa@is.s.u-tokyo.ac.jp)

existing molecules.

Our simulator consists of two main parts, one for solving differential equations and the other for enumerating reactions and generating new molecules. The two parts rely on each other. In particular, the latter part avoids combinatorial explosion by ignoring molecules present in small concentrations, while the former part computes the concentration of each molecule using differential equations.

This paper is organized as follows. In the next section, we give a brief overview of our simulator. In Section 3, the implementation of the simulator is described in detail. Section 4 provides examples of simulations and Section 5 describes PCR (polymerase chain reaction) simulation and parameter fitting for PCR using the Genetic Algorithm (GA). A final section contains the concluding remarks.

2 Overview of the simulator

The simulator we are now developing consists of two main parts. The first part manages molecules in a simulation. It enumerates possible reactions among existing molecules and generates new molecules according to the reactions. Since a molecule in our simulator is a hybrid of abstract sequences (see the next section), the first part is called the *hybrid analyzer.* The other part, called the *numeric calculator,* numerically solves differential equations for reactions found by the hybrid analyzer. The two parts are closely related and rely on each other. They form the core of the simulator.

The hybrid analyzer searches for reaction sites in abstract sequences of existing hybrids and generate new hybrids based on the reaction sites discovered. Since there are a huge number (often an infinite number) of possible reactions in DNA computing, it may generate too many minor products, leading to a combinatorial explosion. It controls the generation of new hybrids by using the concentrations of reactants that generate them. The numeric calculator calculates the concentration of each hybrid from concentrations of initial hybrids by solving differential equations for the reactions enumerated by the hybrid analyzer.

The simulation also contains other modules: a graphical user-interface module and supplementary modules. The user-interface module draws two-dimensional plots whose axes are the concentration of each hybrid and the number of steps. The module utilizes the Java AWT interface so that users can execute it under typical browsers such as Netscape4.5. Users can add new functions to our simulator by preparing supplementary modules. If they are written in the Java language, they can be easily plugged in. An example of a supplementary module is the program for the genetic algorithm discussed in Section 5.

Figure 1 shows the overview of the simulator and summarizes interactions between the modules. Details of each module are described in subsequent sections.

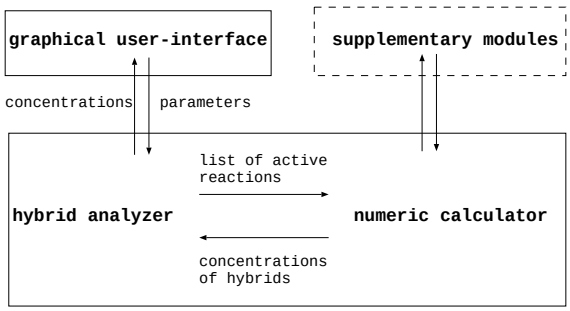

Figure 1: Summary of interactions within the simulator

3 Implementation of the simulator

3.1 Representation of molecules – abstract sequences and hybrids

The main purpose of the simulator we are now developing is to check whether expected reactions dominate other unexpected ones. For this purpose, it is not necessary to simulate real DNA sequences. In order to make simulations efficient, it is preferable to utilize more concise symbolic representation of DNA sequences. We call such symbolic sequences *abstract sequences.*

For example, real DNA sequences "`ATC`", "`CTTAGC`", and "`AAGCCGGAT` can form a hybrid by the "`A-T`" and "`G-C`" pairs as shown on the left side of Table 1. We can represent these real DNA sequences as *abstract sequences* "**X**", "**ZA**", and "**zyx**", respectively, where abstract sequences consist of *abstract bases*, such as **X** and **x**, each representing a real DNA sequence. An abstract base written as a lowercase letter is complementary to the abstract base written as the corresponding uppercase letter. The hybrid consisting of abstract sequences is shown on the right side of Table 1.

In the current implementation, abstract sequences that are not complementary with each other never hybridize. Therefore, mis-hybridization such as one-base mismatch cannot be simulated properly. It would be possible to define similarities between non-complementary bases and simulate reactions with mis-

real DNA hybrid	abstract hybrid
$5' \rightarrow 3'$	$5' \rightarrow 3'$
ATC CTTAGC	**X ZA**
| | | | | |	| |
TAGGCCGAA	**xyz**
$3' \leftarrow 5'$	$3' \leftarrow 5'$

Table 1: Example of a hybrid made of three sequences

hybridization. Even under the current implementation, however, it is not impossible to simulate real DNA sequences by representing real bases as abstract bases, i.e., A as **A**, G as **G**, T as **a**, and C as **g**.

As a data structure, a hybrid consists of a set of abstract sequences, each of which has links to other abstract sequences in the hybrid. Each link represents a hydrogen bond between subsequences in the hybrid. The most difficult issue in the implementation of hybrids is the maintenance of the identity among hybrids. When a new hybrid is created by a reaction, we should check whether it is identical to one of the existing hybrids. In order to check the identity, we always *normalize* a hybrid when it is newly created. By normalization, abstract sequences in a hybrid are topologically sorted according to the bonds in the hybrid. However, if a hybrid contains multiple copies of the same abstract sequence, normal forms of a hybrid may not be unique.

We avoided implementing the identity by means of graph isomorphism, because the simulator can work properly even if the same hybrid is separated into two different hybrids. In this case, although the concentration of the hybrid is also divided into two, we can get the right concentration simply by adding them.

3.2 The hybrid analyzer for abstract sequences

The hybrid analyzer searches for reaction sites in abstract sequences and enumerates possible reactions among existing hybrids. The kinds of reactions currently supported by the simulator are shown in Table 2.

Hybridization is a reaction in which abstract sequences from two hybrids form a complementary pair and hybridize to become a new one. *Self-hybridization* is a reaction in which abstract sequences in a single hybrid form a new complementary pair. *Denaturation* occurs when a bond between abstract sequences in a hybrid disappears. The hybrid may or may not be split into two hybrids.

reaction	formula	tentative parameter
hybridization	$k[H_1][H_2]\Delta t$	$k = 0.1$
self-hybridization	$k[H]\Delta t$	$k = 1.0$
denaturation	$kf^n[H]\Delta t$	$k = 1.0$, $f = 0.5$
digestion by restriction enzyme	$k[H]\Delta t$	$k = 10.0$
extension by polymerase	$(k/n)[H]\Delta t$	$k = 10.0$
concatenation by ligase	$k[H]\Delta t$	$k = 10.0$

Table 2: Reactions supported by the simulator

By *restriction enzyme digestion,* double-stranded abstract sequences, i.e., abstract sequences that form a complementary pair, are cut at the specified restriction site. By *extension by polymerase,* the $3'$-end of an abstract sequence is extended while guided by its complementary sequence. By *concatenation by ligase,* two abstract sequences are concatenated when they share their complementary (bridge) sequence.

As is explained in Section 2, the generation of new hybrids is controlled by the concentrations of the reactants in a reaction. To state more precisely, the simulator is given a constant concentration as a *threshold.* For each reaction the hybrid analyzer enumerates, it is checked whether all of its reactants have a concentration greater than the threshold. If they do, the reaction is made active and sent to the numeric calculator described below.

3.3 The numeric calculator based on differential equations

The numeric calculator numerically solves differential equations by a standard method. The equations are simply based on the usual chemical reaction kinetics. Chemical reactions can be described by the following equations. If there is only one reactant in a reaction, the equation is:

$$\Delta[Product] = k[Reactant]\Delta t,$$

where $\Delta[Product]$ is the increase in the concentration of the generated product, k the rate constant of this reaction, $[Reactant]$ the concentration of a reactant, and Δt the time interval. If there are two reactants, the equation is:

$$\Delta[Product] = k[Reactant_1][Reactant_2]\Delta t$$

In a simulation, $\Delta[Product]$ is calculated step by step and the concentrations of the reactants and the product are updated accordingly. Table 2 contains the formula for $\Delta[Product]$ and the tentative values for each reaction. For *denaturation, n* in the formula denotes the number of abstract bases to denature.

In the current implementation, one step of the numeric calculator is followed by the execution of the hybrid analyzer, which recomputes active reactions according to the updated concentrations. Then another step of the numeric calculator is executed with the new active reactions.

3.4 User-interface

The graphical user-interface draws a graph of the concentration of each hybrid. Since both the user interface module and the simulator are written in Java, it was easy to make a Java applet for them. Figure 2 is a sample image of our simulator's applet. This shows the result of the simulation described in Subsection 4.1. Notice that at each step of the simulation, the ten largest concentrations are plotted. Users can enter the initial values for the parameters. The applet is now accessible through our project's web server.[1]

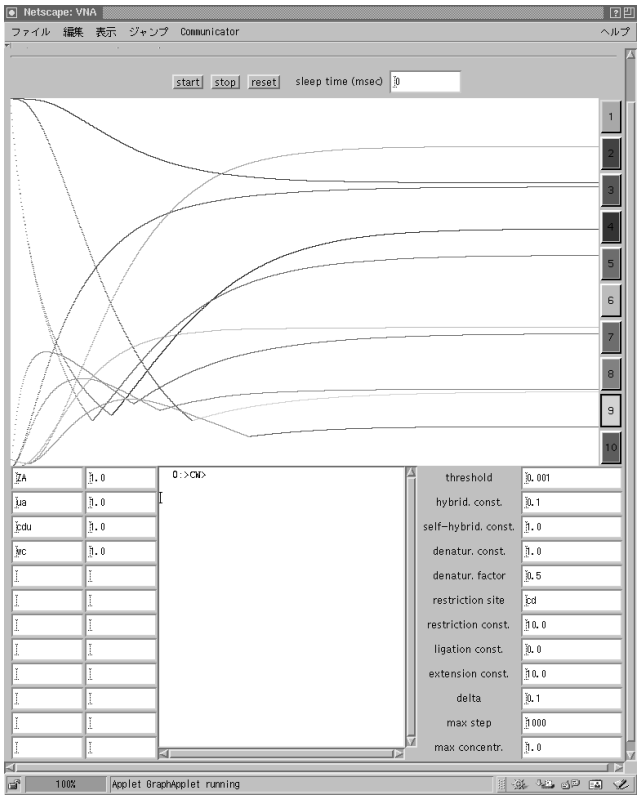

Figure 2: Example image of the applet

4 Examples of simulations

After we developed a prototype of our simulator, we examined it by simulating some practical applications: 1) computation of Boolean circuits [3], 2) formation of DNA tiles [9], and 3) PCR. This section describes 1) and 2), while the simulation of PCR is described in the next section, because it involved using the genetic algorithm to find the proper values for PCR parameters.

4.1 Computation of Boolean circuits

Ogihara and Ray proposed a new DNA algorithm for simulating a bounded fan-in Boolean circuit [3]. Their algorithm depends on a cascade of extension reactions, in which newly generated hybrids are used to generate further hybrids. Since the protocol of the algorithm is sophisticated and rather complex, it is important to verify the feasibility of the algorithm.

In Ogihara and Ray's algorithm, inputs to and outputs from a Boolean circuit are translated into single-stranded (abstract) sequences. A Boolean operation executed in one layer is implemented as two reactions of DNA computing: *polymerase extension* of primers, and *restriction enzyme digestion.* In the first reaction, input sequences of the layer play the role of primers, i.e., they hybridize and are extended by polymerase, guided by each other. Since the extended hybrids are double-stranded and contain a restriction site, they generate new short sequences when cut by a restriction enzyme. These become the output of this layer and they are sent to the upper layer of the circuit as input. This process is repeated to generate output sequences for the entire circuit.

In this simulation, the sequences listed in Table 3 were put to the virtual tube. After 1,000 steps, we got the final product, which is shown on the fourth row of Table 3[2] . The concentrations of the initial sequences are also given in the figure. The parameters for reactions are those listed in Table 2. We used 0.001 as the threshold of concentrations. Each time interval was 0.1. The only restriction site is in between **c** and **d**.

[1] The URL of the simulator is `http://nicosia.is.s.u-tokyo.ac.jp/moco/vna/example1.html`.

[2] Abstract sequences **ZA**, **ua**, **cdu** and **wc** in this paper correspond to sequences A, B, unti-E anti-U, and anti-W anti-E (not all anti-E, but only half of anti-E) in Figure 1 of Ogihara and Ray's paper [3], respectively.

initial sequence	initial concentration	major product	final concentration
ZA	1.0	**c**	0.87
ua	1.0	**wc**	0.78
cdu	1.0	**uaz**	0.77
wc	1.0	**CW**	0.64
		duaz	0.58
		ZAUD |||| **zaud**	0.38

Table 3: Simulation of Ogihara and Ray's algorithm

In the course of testing the prototype of the simulator, we first had the hybrid analyzer enumerate all the possible hybrids of this algorithm. The number of different hybrids exploded immediately, because an infinite number of hybrids can be generated by this algorithm. After we combined the numeric calculator with the hybrid analyzer and introduced the concentration threshold to prune minor products, we obtained the desired result.

4.2 Formation of DNA tiles

A DNA tile is a complex hybrid made of four DNA sequences. Winfree et al. performed some experiments by using DNA tiles. They made DNA sheets with a controlled pattern out of DNA tiles [9]. They even proposed a method to simulate two-dimensional cellular automata by DNA tiles. Although DNA tiles were already synthesized in a laboratory, we tried to simulate the formation of DNA tiles by our program. Figure 3 shows the DNA tile and its DNA sequences.

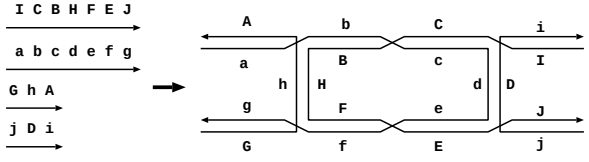

Figure 3: Winfree's DNA tile

Initially, we tried to simulate the formation of DNA tiles using the method that we used for Ogihara and Ray's algorithm. The final DNA tile has eight bonds, but our best result was a hybrid that contains six bonds (Figure 4). The simulation failed because the number of intermediate hybrids is combinatorially huge and the concentrations of these hybrids are extremely small, although the final DNA tile might exist at a reasonable concentration. A large threshold will eliminate reactions leading to the final DNA tile, while a small threshold results in the combinatorial explosion.

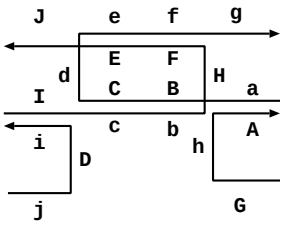

Figure 4: Incomplete DNA tile

Instead, we introduced a stochastic model, where each reaction is made active according to the probability proportional to the value of $\Delta[Product]$. The refined simulator almost always generated a path from initial sequences to the final DNA tile, although not 100% of the time. The stochastic approach showed a possibility to compensate the inability of the the threshold approach to find a reaction path.

5 Fitting parameters for PCR by GA

We also tried to integrate simulating DNA computations and fitting parameters by the genetic algorithm (GA). For simulating DNA computations, we must tune many parameters, such as rate constants and factors. GA can be a good method for fitting such parameters to real *in vitro* experimental values. However, we have no experimental data which are suitable for the target of parameter fitting at the moment.

In this paper, in order to demonstrate the usefulness of GA, we report the result of fitting parameters for PCR using GA. The parameters for simulations, such as rate constants and factors, are hypothetical.

We tried to apply the simulator to PCR experiments. PCR is a standard method used to amplify a template DNA sequence by using initiator sequences called *primers*. PCR involves thermal cycles. Hybridization between template sequences and primers and polymerase extension both require relatively low temperatures, while denaturing the template sequences requires a higher temperature. For PCR experiments, we must decide four parameters at least, i.e. the low temperature and the high temperature, the length of the period of the low temperature, and that of the high temperature. We tried parameter fitting of these four

reaction	formula	parameter
hybridization	$kn[H_1][H_2]\Delta t$	$k = 0.1$
self-hybridization	$kn[H]\Delta t$	$k = 1.0$
denaturation	$kf^n[H]\Delta t$	$k = 9.0$, $f = e^{-24.1/T+19.6}$
extension by polymerase	$(k/n)[H]\Delta t$	$k = 5.0$

Table 4: Reactions in PCR

parameters by adding a supplementary module for the genetic algorithm.

Parameters used in the PCR simulation are shown in Table 4. In the table, T denotes the temperature ranging from 0.9 to 1.3. We used 0.0001 as the threshold of concentrations and each time interval was 0.1.

5.1 Method

Genetic algorithms (GAs) are powerful methods for optimization and have been applied to many fields. Some of them can be used to fit real-valued parameters directly. Several ideas have been proposed to implement recombination operations in real-coded GAs like one point crossover in string coded GAs. UNDX (unimodal normal distribution crossover) is known powerful in high dimensional epistatic functions [4]. We used BLX (blend crossover) [2], since it is known a little less powerful than UNDX but extremely easy to implement. There also have been proposed a lot of generation alternation models. We combined BLX into MGG (minimal generation gap) model, since it is also easy and is known to be a robust first choice [5]. As for genes that go outside the domain of parameters, we followed Tsutui's guideline for mirroring [7].

The population is 100. In each generation, 20 children are generated from two parents. The parents are then replaced by the best one among the parents and 20 children, and a randomly selected one. The concentration of the template sequence after a 1,000-step simulation by our simulator was used as the evaluation function.

5.2 Result

Figure 5 shows the result of 5,000 generations. Figure 6 is the graph drawn by the simulator with the best parameters found using GA: the low and high temperatures are 0.90 and 1.23, whose periods are 22 and 1 steps, respectively. As the temperature 1.0 is intended to represent 300K, the temperatures 0.90 and 1.23 correspond to 270K and 369K, respectively. The increasing curve is that of the concentration of the amplified template, including single-stranded and double-stranded sequences. The result indicates the feasibility of the integration of the simulation of DNA computing and the parameter fitting by GA, even if it is based on a very simplified model. The result should be compared with that of *in vitro* experiments and the reaction model on which the simulator is based should be refined.

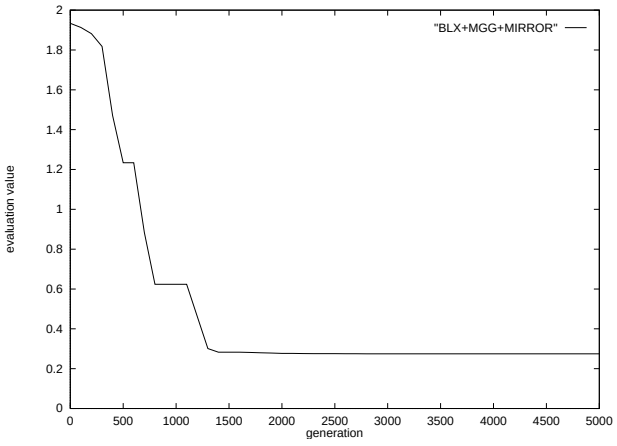

Figure 5: Evaluation value of the best individual in each generation

6 Discussion on the Limitations of the Simulator

We examined some simulations of DNA computing. Most of the simulations were based on the threshold approach. This approach is more important than the stochastic approach, because it can compute real number concentrations of hybrids. However, we have found some limitations of the threshold approach. The simulation of the DNA tile formation from four DNA fragments which we reported was based on the stochastic approach. Of course, the simulation was also examined by the threshold approach, whose result was not successful. This is because there exist a huge number of reaction paths to the final DNA tile from the initial fragments via intermediate hybrids each of which has a very small concentration, and the threshold prevents all the reaction paths towards the DNA tile. In such cases, simulations based on the threshold approach cannot work properly.

The other problem of the threshold approach is that we cannot decide the proper threshold value before a

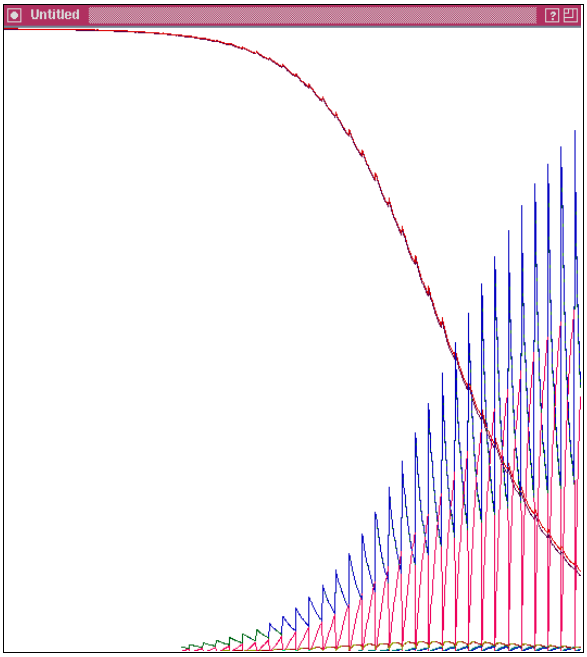

Figure 6: PCR simulation with the best parameters

simulation. In the case of Ogihara and Ray's Boolean circuit computation, we repeated simulations until we found the proper threshold value. When we set the threshold under that value, we got almost the same results. However, simulation results changed according to the value of the threshold in some cases. The threshold in such cases might not be small enough. The theoretical analysis was not made yet.

7 Concluding Remark

We constructed a simulator for DNA computing and tried parameter fitting by using GA methods. Although our system is not complete yet, we successfully demonstrated the possibility of simulating DNA computing. Many issues remain for future investigation. The correctness of our threshold approach should be examined experimentally as well as theoretically. As the system is based on abstract sequences, we must also establish a relationship between real DNA sequences and abstract sequences.

Acknowledgement

We thank Prof. Akira Suyama (University of Tokyo) for valuable comments on simulating DNA experiments. This research was supported by the Japan Society for the Promotion of Science under *Research for the Future* Program (JSPS-RFTF 96I00101).

References

[1] Dennis Bray, Robert B. Bouret and Melvin I. Simon: Computer Simulation of the Phosphorylation Cascade Controlling Bacterial Chemotaxis, *Molecular Biology of the Cell*, Vol.4, pp.469–482, 1993.

[2] Larry J. Eshelman and David J. Schaffer: Real-Coded Genetic Algorithms and Interval-Schemata, *Foundation of Genetic Algorithms 2,* pp.187–202, 1993.

[3] Mitsunori Ogihara and Animesh Ray: DNA-Based Self-Propagating Algorithm for Solving Bounded-Fan-In Boolean Circuits, *Genetic Programming 98,* pp.725–730, 1998.

[4] Isao Ono and Shigenobu Kobayashi: A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover, *Proc. ICGA'97,* pp.246–253, 1997.

[5] Hiroshi Satoh, Masayuki Yamamura and Shigenobu Kobayashi: Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation, *Proc. IIZUKA'96,* pp.494–497, 1996.

[6] Masaru Tomita, et al.: E-CELL: Software Environment for Whole Cell Simulation, *Bioinformatics,* Vol.15, No.1, 1998.

[7] Shigeyoshi Tsutsui: Multi-parent Recombination in Genetic Algorithms with Search Space Boundary Extension by Mirroring, *PPSN98,* 1998.

[8] Erik Winfree: Simulations of Computing by Self-Assembly, *Preliminary Proc. of Fourth International Meeting on DNA Based Computers,* pp.213–239, 1998.

[9] Erik Winfree, Xiaoping Yang and Nadrian C. Seeman: Universal Computation via Self-assembly of DNA:Some Theory and Experiments, *Preliminary Proc. of Second Annual Meeting on DNA Based Computers,* pp.172–190, 1996.

[10] Byoung-Tak Zhang and Soo-Yong Shin: Molecular Algorithms for Efficient and Reliable DNA Computing, *Genetic Programming 98,* pp.735–742, 1998.