Improving Sequence Design for DNA Computing

Masanori Arita
arita@etl.go.jp
Supermolecular Science Division
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba-shi
305-8568 Ibaraki, Japan

Akio Nishikawa and Masami Hagiya
{nisikawa,hagiya}@is.s.u-tokyo.ac.jp
Department of Infomation Science
Graduate School of Science
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
113-0033 Tokyo, Japan

Ken Komiya, Hidetaka Gouzu and Kensaku Sakamoto
Department of Biochemistry
Graduate School of Science
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
113-0033 Tokyo, Japan

Abstract

Good design of sequences is a necessity for successful DNA computing, but how to achieve this 'goodness' is unknown. We point out the importance of adjusting the composition of sequences to fit each computational model, and introduce two new sequence generators. The first, which uses genetic algorithm (GA), exhibits several limitations in its ability to keep specified constraints on sequences. For this reason, we designed the second generator using a random generate-and-test algorithm. We discuss why the use of the latter generator, with its easy-to-use user interface, was necessary to optimize the multiple set of encoding constraints required in DNA computing.

1 Introduction

Currently, no multi-purpose library of sequences has been established for DNA computing. This absence may be explained by noting the large diversity of DNA-based computational architectures. Given the differing enzymatic and experimental requirements of each architecture, it seems indeed unlikely that we can find a single set of DNA sequences which effectively caters to the fidelity requirements of every architecture. As a result, the need to design and synthesize a new set of sequences for each new laboratory experiment appears to be unavoidable. In order to ease this design process, what is needed is a DNA sequence design tool which can adapt to the changing needs of the various DNA architectures.

Many rules of thumb have been used in sequence design. One particularly common method of preventing unwanted modes of hybridization is to minimize the similarity of the various sequences, by using a convenient similarity measure. The use of sequences with uniform melting temperatures has also been suggested to be useful in establishing uniformity among the reaction rates of the various planned hybridizations. If restriction enzymes are to be used, it is also important to prevent the unplanned formation of double stranded DNAs containing the corresponding restriction site. Additional constraints may also be necessary, depending on the computational architecture employed.

There is no well established method for designing sequences which satisfy such multiple sets of encoding constraints. The development of a general method is again complicated by the different needs of the various DNA-based computational architectures.

Moreover, a design heuristic which is well established in biotechnology may cause problems when applied to DNA computing. For example, the GC content is known to be a good indicator of the melting temperature of double stranded DNAs. As a result, PCR primers are typically selected to have the same GC content, in order to allow for the control of fidelity by using a stringent reaction temperature. In DNA computing, on the other hand, the requirement that sequences be of uniform GC content drastically narrows the combinatorial variety of the sequences which may be used. As a result, there is a clear trade-off between the ability to enforce computational fidelity by means of experimental stringency and the size of the problem instance which can be solved. The quantitative nature of this trade-off, however, is unknown.

At minimum, a good sequence generator should satisfy the following constraints. First of all, it must be flexible enough to handle the various multiple constraints imposed by existing DNA architectures. Secondly, it must be able to accept partially-determined input sequences, so that a user can set restriction sites at planned positions. Thirdly, its limitations must be well-defined, in order to prevent futile searches for better sequences.

In this paper, we stress the importance of adjusting the constraints for sequence-design to fit the computational model. In addition, we introduce two sequence generators which are flexible enough for general sequence design. Finally, this paper compares two strategies for the design of sequences: the use of a GA and the use of a random algorithm.

The GA is probably the most popular method of parameter optimization for a problem which is difficult to mathematically formalize. Therefore, we first implemented a generator which uses a GA for sequence design, and then tested the output sequences experimentally. As the number of design constraints increased, however, we encountered difficulties in setting the fitness evaluation function. Defining an appropriate fitness function was particularly complicated by the fact that the relative importance of each constraint was unknown. For this reason, we designed the second generator which utilized a random generate and test algorithm, allowing users to set an explicit, independent threshold value for each constraint. This paper suggests that the GA-based generator is not appropriate for optimizing multiple parameters, and that the random generator is more practical for the design of good sequences.

The paper is organized as follows. In Section 2, we briefly describe related work and introduce the background of this research. In Section 3, we introduce the two programs implemented for the design of sequences for whiplash computation model. Laboratory results are presented in Section 4. Both a discussion of the results and suggestions for future work are presented in Section 5.

2 Background

In the selection of primers for a PCR experiment, the most important considerations are the use of sequences with uniform GC content and the avoidance of sequences which are substantially self-complementary. Sequence design for DNA computing has more freedom than PCR primer design, since DNA computing does not use natural DNA. On the other hand, the sequences used in DNA computing will often be extended by polymerase or by ligase, and additional design constraints will often arise due to these modifications.

In a normal primer-design tool, the secondary structure of candidate primers is not computed in detail. Instead, all primers which satisfy a particular set of design criteria are output. A laboratory expert then selects one or more primers from amongst the many candidates suggested by the computer program. This methodology is also typical in DNA computing. Since it is difficult to precisely estimate the exact molecular interactions which each strand will experience during the computation, the expert must intuitively pick one set of 'good-looking' sequences from among many candidates. In this section, we shall summarize the basic constraints used to select sequences, and describe how an expert selects a good set of sequences for DNA computing.

2.1 DNA code-oriented Constraints

Marathe *et al.* interpreted the constraints on DNA sequences from the perspective of combinatorial code design, and proposed to use the following set of design constraints [7]:

- Hamming constraint $\cdots$ A large Hamming distance should be kept between any two sequences.
- Reverse-complement constraint $\cdots$ A sequence should not hybridize with the reverse-complement of other sequences.
- Reverse constraint $\cdots$ A sequence should not hybridize with the reverse of itself and of other sequences.

Deaton *et al.* experimentally demonstrated the goodness of Hamming constraint [1, 2], and they further proposed the following more elaborate measure.

2.2 H-measure

Garzon *et al.* introduced the H-measure, which is essentially the minimum of the Hamming distances calculated by shifting one sequence against the other [4]. The advantage is that this measure considers the frame-shift of sequences. The definition of H-measure is as follows. Readers are referred to their original paper for detail.

> *The* **H-measure** *between two n-mer oligos x, y is*
>
> $$|x, y| := \min_{-n<k<n} H(x, \rho^k(y)),$$
>
> *where ρ (ρ^{-1}) is the right (left) shift and $H(*, *)$ is the ordinary Hamming distance.*

In this paper, however, we use the Hamming constraint, not H-measure, since the Hamming constraint was reported to be effective in laboratory experiments [1].

Marathe *et al.* also proposed the melting-temperature and free-energy constraints, which are obtained as the sum of weights associated with DNA 2-mers (16 patterns in total). These constraints are intended to reflect the physico-chemical characteristics of DNAs, and are different from the combinatorial methodologies. In fact, such physico-chemical aspects are more essential (and therefore important) in the design of sequences. In the following, we shall list such important aspects, which tend to be overlooked by theoretical scientists.

2.3 GC content

In the selection of PCR primers, sequences are chosen to have similar GC content, because it is a good indicator of the melting temperature of double stranded DNAs. According to the isothermal model [11], it is therefore best to set the GC content of all sequences to be *exactly* same.

Generally in the design of sequences, if we are allowed to use either `G` or `C`, it is better to choose `C`. Since the base-pair `G-T` is almost as stable as `A-T` pair, and since `G-T` pair does not skew the double-helical structure of DNAs [10], `G` can form a base-pair with `T` in the same way as `A` does. If we put many `G`s in the sequences, therefore, the likelihood of unplanned hybridization will increase. On the other hand, `G` is necessary for increasing the combinatorial variety of sequences.

2.4 Hybridization at Termini

One of the most important constraints is the avoidance of unwanted extension by polymerase or by ligase. When the 3'-end of a sequence hybridizes with a 'false' site and forms a double stranded region, unwanted extension of DNA may start from the 3'-end in the presence of DNA polymerase. The longer the length of the double stranded region, the more efficiently the 'priming' of DNA synthesis becomes.

Therefore, one design criterion is to avoid complete hybridizations of a certain length, especially at the 3'-end. From a combinatorial perspective, this constraint is interpreted as avoiding the occurrence of terminal n bases in the midst of other sequences. In our work, we consider $n = 6$ to be a critical case by the following reasoning. Note that the terminal subsequence should not appear in the enzymatically extended or connected parts of sequences.

Consider a DNA sequence S of length $n = size(S)$, where n is an even number (i.e. $n = 2m$), for convenience. Given with 50% GC content, m positions in S will be either `G` or `C`, with the remaining m positions either `A` or `T`. The number of possible sequences of length n and of 50% GC content is $N_n = {}_{2m}\mathrm{C}_m 2^m \times 2^m$. The number N_n is an upper bound of the longest possible size of a circular sequence, which contains no duplicate occurrence of any subsequence of length n.[1] Let us suppose that we use oligomers of length l in DNA computing. We call these short oligomers as *units.* If these units are to be circularly ligated, under the condition that no duplicate n-mers appear in the total circular sequence, then N_n/l bounds the maximum number of units which can be ligated. Let us suppose that we need k units for our experiment. If $k > N_n/l$, then even when all units are optimally designed, $k * l - N_n$ duplications of n-mers will occur, if all k units are circularly concatenated. Since each unit has two termini, $2k$ patterns of n-mers represent terminal sequences. The probability of matching a duplicate n-mer with a terminal sequence is roughly $1-(2*k/N_n)$. This value is not so trivial when n is a small number.

Table 1 shows some results of this estimation. The probability of the complete match of a n-mer becomes smaller as n gets larger. On the other hand, a completely hybridized sequence of length 6 can initiate the priming in PCR, and may induce erroneous polymerization from the 3'-end. Considering this trade-off and our problem size, we consider $n = 6$ to be the critical number for the size of our sequence design. Hereafter, the complete hybridization of length n is denoted *n-complete hybridization.*

---

[1] In fact, this circular sequence does *not* exist if we restrict the GC content to be exactly 50%. However, we assume its existence only to give an upper bound of the number of units. Note that however, its inexistence is not proven if some range in the GC content is allowed. When an arbitrary GC content is allowed, the circular sequence is called a De Bruijn sequence, and the algorithm for its generation is studied extensively [3, 8].

Table 1: Table of n and corresponding N_n

n	2	4	6	8
N_n	8	96	1280	26880

2.5 Summary of Constraints

In summary, the constraints to be considered in sequence design are as follows.

Constraints from Problem Design
Restriction sites should appear only at planned positions.

Constraints for Reaction Efficiency
The GC content of sequences should be uniform, minimizing the amount of `G`. In addition, a large Hamming distance should be kept among units.

Constraints for Error Avoidance No hybridization of length 6 should occur at the termini of units. This constraint should be checked for all the sequences (and their reverse-complementary sequences if the units become double stranded) to be designed, and for their possible enzymatic extensions.

3 Design Strategies

The sequences we designed are intended to be used for solving NP-complete problems with the whiplash PCR model, in which each single stranded DNA reacts intramolecularly to simulate a state-machine [6]. In this model, one single stranded DNA can simulate one machine, by polymerizing its 3'-end in each hybridization step (Figure 1).

Details regarding the implementation of NP-complete problems using the whiplash PCR model have been omitted from this paper. Interested readers are referred to our other works [6, 11]. Figure 2 shows the DNA sequence for an experiment using the whiplash model. The units to be designed are defined as in Table 2.

3.1 Design with GA

In order to process DNA using a GA, each nucleotide is represented by two bits. The DNA sequences are then translated into binary sequences. Each set of 12 sequences in Table 2 is considered one gene by the GA, and each set of genes is passed to GENESIS library [5]. Important parameters for GENESIS are shown in Table 3. The score w for a given gene is computed as fol-

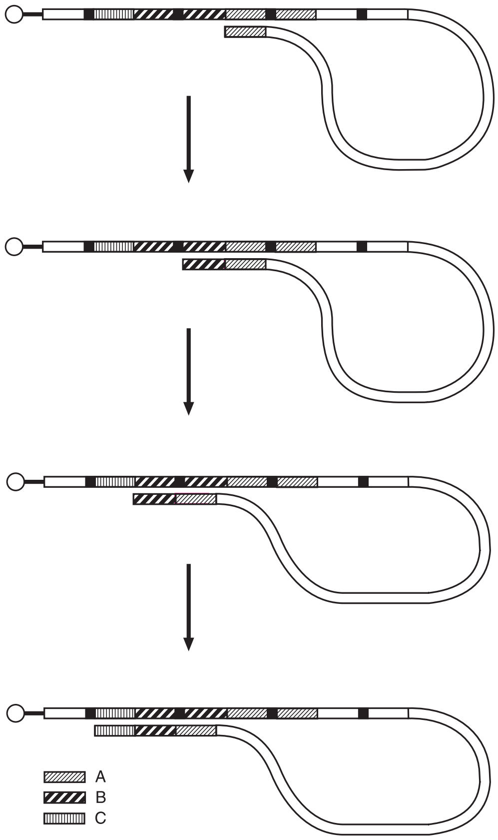


Figure 1: Whiplash Model. The 3'-end (unit A) hybridizes with its complementary sequence and polymerizes its terminus by one unit (unit B). After separation, the 3'-end (unit B) hybridizes to a different position, and polymerizes the terminus by another one unit (unit C). Each extension is terminated by a special 'stopper' sequence shown as black units.

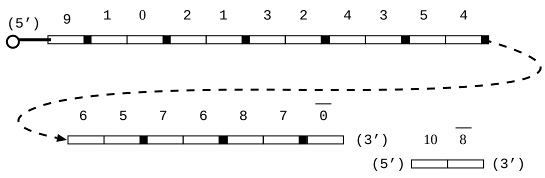


Figure 2: Overview of the target DNA. Each number corresponds to one unit, and the bar '—' indicates a reverse-complementary sequence. A black part is the 'stopper' sequence, which is `AAA` in our implementation.

Table 2: Definition of DNA Units. Units with a dagger (†) are designed with only three bases (C, G, and T).

no.	size	no.	size	no.	size
$0^\dagger$	15	$4^\dagger$	15	$8^\dagger$	15
$1^\dagger$	15	$5^\dagger$	15	9	20
$2^\dagger$	15	$6^\dagger$	15	10	20
$3^\dagger$	15	$7^\dagger$	15	$11^\dagger$	20

lows. A smaller w indicates a better set of sequences. The set of sequences in a gene is denoted S. The symbols $k_1 \sim k_4$ denote user-defined variables.

Score for GA

1. If the sequence contains restriction sites at a 'false' site, $w = +\infty$.
2. Compute the distribution of the GC content of units.

$$w = k_1 * \sum^{S} (\mathrm{GC} - \mathrm{GC}_{user\_defined})^2$$

3. Reward larger Hamming distance.

$$w = w - k_2 * \sum_{s,t \in S} \text{Hamming distance}(s,t)$$

4. Penalize repetition of the same base.

$$w = w + k_3 * (\text{occurrence of TTT or AAA})$$

5. Penalize complete hybridization at the 3'-end.

$$w = w + k_4 * (\text{maxsize of complete hybridization})$$

Table 3: Parameters for GA

population size	100
crossover rate	0.6
mutation rate	0.001

The advantage of using the GA is that fitness scores are continuously rated. Even when we do not know the exact threshold values for each parameter, the GA is expected to output better sequences on average. For example, it is usually difficult to estimate the largest possible Hamming distance between the units to be designed. The GA can find, however, the nearly optimized sequences automatically. On the other hand, the following disadvantages of the GA were observed.

The trade-off between parameters is hard to control using a GA. In Section 2 it was mentioned that G should be avoided in sequence design. Since most sequences must be designed with only three bases in the whiplash model, sequences tend to evolve using only two bases (T and C). This tendency makes it difficult to increase the Hamming distances between sequences. If G is included, however, the possibility of self-hybridization increases, because the possibility of forming a G-C pair increases.

How to regulate this trade-off inside GA depends on the relative scaling of parameters. Although four variables $k_1 \sim k_4$ in our GA fitness evaluation function heavily affect GA results, there is no good indicator for adjusting these parameters.

In a general GA, all parameters must be calculated into a single 'fitness' score. Therefore, if the relative importance of the parameters is not known, control of the GA output will be very hard. We tentatively set the four parameters to be $k_1 = 1, k_2 = 5, k_3 = 10$, and $k_4 = 10$, but these values were empirically determined and are not based on any theoretical basis.

The polymerization (or ligation) pattern of units is laborious to represent using a GA. When units are polymerized or ligated, all their connected parts should be checked for the possibility of hybridization, of short Hamming distance, and of restriction sites. Since each experiment uses a different extension pattern between units, much effort is required to correctly implement these constraints in a GA fitness function. It is also laborious to define a new function for each polymerization pattern.

3.2 Design with Generate-and-test Algorithm

To overcome the difficulties of sequence design with a GA, we designed another generator in which threshold values for each parameter can be explicitly defined. The inputs to the generator are as follows.

- Definition of units' size to be designed.
- The pattern of units' extensions.
- Subsequences which should not appear.

A user can pre-define and specify a portion of the unit sequences, allowing restriction sites to be set at planned positions. (These pre-defined sequences are excluded from the check for forbidden subsequences.) In this generator, the sequence design is iterative: each unit is assigned a randomly generated sequence until

Table 4: Algorithm for the iterative generation

1. Pick up one unit whose sequence is not yet fully designed. If all units are designed, end this program.
2. Fill the unit with random bases. (See below for detail.)
3. Discard the sequence if the filled unit or its connection with other units does not satisfy the following restrictions. Parameters r, n, and h are user-defined.
   - The same base should not repeat more than r times.
   - The unit should not n-completely hybridize with itself.
   - The Hamming distance between the unit and itself should be more than h.
   - The unit should not n-completely hybridize with previously generated sequences or with connected parts of generated sequences.
4. Go to Step 1.

the sequence passes the test for its appropriateness. The algorithm is shown in Table 4.

Let us assume that the generator is acting to design a unit S in Step 2. The generator first determines positions of either `G` or `C` inside S randomly, according to the GC content given by the user. (The remaining part will be assigned with either `A` or `T`.) Next, the generator randomly chooses one of two bases for each position. In this way, the generator can precisely regulate the GC content while retaining randomness.

In this method, the first unit is the easiest to generate. The design then becomes increasingly difficult as the number of generated units increases. Although it is difficult to estimate the running time of the generator, it has the following advantages.

- parameters can explicitly be set for each sequence.
- sequences can easily be (separately) designed.

The disadvantage is that parameters should be appropriately set before the execution of the generator, otherwise the generator cannot find sequences satisfying the given constraints. Moreover, the generator cannot suggest the possibility of a better set of sequences satisfying the same constraints. However, if we know the parameters to be satisfied beforehand, this method is efficient and useful.

All the operations from setting parameters to redesigning, checking, and printing of the units under design are executed through a command-line user interface.

4 Experimental Results

The sequences shown in Table 5 were designed using the GA generator. In our laboratory experiment, one additional `T` is attached at the end of each sequence in order to use Taq polymerase, because this enzyme tends to attach an additional `A` after the polymerization.

Figure 3 shows the experimental result of the transition steps. The initial five extensions were observed to be successful, but the final two steps failed. The efficiency in the extension appeared to decrease as the whiplash reaction proceeded (from lane 1 to 6). In addition, we observed many bands that were shorter than the expected length. Although the decrease in efficiency may be due to the nature of the reaction itself, the unwanted bands were assumed to be generated by mishybridization between the designed DNA sequences.

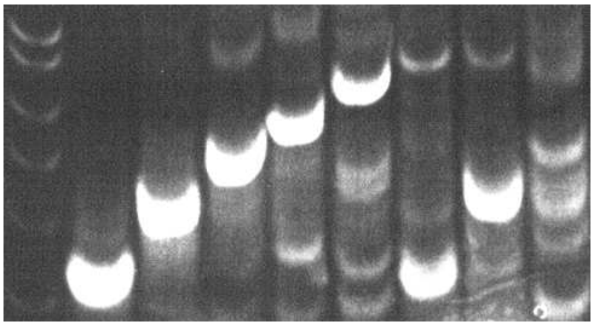
In order to investigate this failure, we picked up three sequences from the unsuccessful bands, and sequenced them.

Table 5: DNA Units output by GA generator

no.	sequence
0	CCGTCTTCTTCTGCT
1	TTCCCTCCCTCTCTT
2	CGTCCTCCTCTTGTT
3	CCCCTTCTTGTCCTT
4	TGCCCCTCTTGTTCT
5	CTCCTCTTCCTTGCT
6	CTTCTCCCTTCCTCT
7	CCTTCCTTCCCTCTT
8	TCCCCTTGTGTGTGT
9	GAGAGAGAGGCCCCCTATCC
10	GAAGAGAAGGGCACCCCTCC
11	GTGGTGTTGCGTCCCTTCCC

The three sequenced fragments contained sequence 5, 6, 2, 1, and 0. By analyzing their sequences, we obtained the following observations.

Sequence Design with only two bases (C,T) is dangerous. In our design, sequences 1, 6, and 7 are



1 2 3 4 5 6 7

Figure 3: Amplified products from the stepwise polymerization reactions. The ladder-forming bands from 1 to 5 show the successful extensions for five steps. The sixth and the seventh extensions were unsuccessful with these sequences.

designed without `G`. Although this design has the advantage that self-hybridization is unlikely to occur, it seems to have increased failure. Probably, the 3'-ends of units make 'false' hybridization during the extension steps due to their sequence similarity.

A large Hamming distance is important. Although complete hybridization of length 7 or larger is avoided by our design, the Hamming distance between units is sometimes small (around 5). The Hamming distance affects the efficiency of reactions as described in Section 1. Short Hamming distance may be the cause of the two failing transitions, following the initial successful ones.

These indicate that the sequences, evolved using the GA, were probably not good enough to avoid accidental mishybridization. Although we know the points to be improved, as described above, our GA generator is not convenient for this purpose. For example, when we want to increase the ratio of base `G` in the designed sequences, it is difficult to know which parameters to weight more heavily, or which are to weight less. In order to utilize experimental results for future sequence design, our random generator may be useful. We are presently testing its validity in ongoing experiments.

5 Concluding Remark

Two sequence generators were implemented, one using a GA and the other using a random algorithm.

The GA generator is useful, when the appropriate threshold values for parameters are not clear. However, it is difficult to reflect user preference by following the explicit setting of parameters, because all parameters must be calculated into a single score for evaluating the set of generated sequences.

The random generator, on the other hand, is flexible and good at designing sequences, when thresholds of parameters are pre-defined, and when a part of sequences are pre-defined. However, if appropriate parameters are unknown, the generator cannot find good sequences.

Given the advantages of both generators, our current design strategy is as follows.

1. Using the GA generator, estimate how critical each constraint is for the planned sequence design.
2. Using the random generator, design sequences which satisfy the user's preference, such as presetting restriction sites or terminal sequences.

The current problem with the random generator is that there is no theoretical guarantee that the random generator can find a set of sequences as good as the one found by a GA, in about the same running time. The theoretical analysis and the efficient implementation of the generator is the main ongoing work.

Also important is the consideration of thermodynamics of reaction [9]. The constraints from thermodynamics are more essential for evaluating the goodness of sequences, but the best way to integrate the physicochemical parameters into the methodology for designing new sequences is not well known. Its integration represents additional future work.

Acknowledgment

The authors thank Dr. John Rose for helpful comments and for corrections of our manuscript. This work is supported by the Japan Society for the Promotion of Science "Research for the Future" Program (JSPS-RFTF 96I00101). The work is also supported by Grant-in-Aid for Scientific Research on Priority Area "Genome Informatics," from Ministry of Education, Science, Sports and Culture, Japan.

Appendix

The sample specification file for our Random generator. The random generator will be publicly available at our FTP site soon.

```
/*   Definition of sequences   */
/* 5' -> 3' from left to right */
/* Seq length by the number of space. */
```

```
/* # number uses CGT only. */
/* $ number uses all ACGT. */
set #0 "                ";
set #1 "                ";
set #2 "                ";
set #3 "                ";
set #4 "GCTTT           ";
set #5 "                ";
set #6 "                ";
set #7 "                ";
set #8 "                ";
set $9 "                     ";
set $10 "                    ";
set #11 "                    ";

forbid GAATCC;
forbid GCGGCCGC;
forbid AAGCTT;
forbid GGATCC;

/* Definition of sequence connection */
/* The number should appear in the above. */
/* ~ is the complementary sequence. */

connect $9 AAA #1 #4 AAA #2 #1 AAA #3 #2 AAA #4 #3
AAA GAATCC AAA #13 #4~ GGCCGC #0 #8 AAA GCTT AAA
#5 #4 AAA #6 #5 AAA #7 #6 AAA #8 #7 AAA GGATCC $10;
```

References

[1] Deaton, R., Garzon, M., Murphy, RC., Rose, JA., Franceschetti, DR., and Stevens Jr, SE. "Reliability and Efficiency of a DNA-Based Computation," *Physical Review Letters*, 80:417–420, 1998.

[2] Deaton, RJ. and Garzon, M. "Thermodynamic Constraints on DNA-based Computing," *Computing with Bio-Molecules: Theory and Experiments,* ed. G. Paun, Springer-Verlag:Singapore, pp. 138–152, 1998.

[3] Fredricksen, H. "A Survey of Full Length Nonlinear Shift Register Cycle Algorithms", *SIAM Review* 24(2):195–221, 1982.

[4] Garzon, M., Neathery, P., Deaton, R., Murphy, RC., Franceschetti, DR., and Stevens Jr, SE. "A New Metric for DNA Computing", *Proc 2nd Ann Genetic Programming Conf*, 472–478.

[5] Grefenstette, JJ.: genesis 5.0 source code. *URL http://www.aic.nrl.navy.mil:80/galist/src/*, 1994.

[6] Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., and Yokoyama, S. "Towards Parallel Evaluation and Learning of Boolean mu-formulas with Molecules". *DIMACS Series in Discret Math and Theor Comput Sci*, 48:57–72, 1999.

[7] Marathe, A., Condon, AE., Robert, MC. "On Combinatorial DNA Word Design", *DIMACS Series in Discret Math and Theor Comput Sci*, 44:75–87, 1999.

[8] Ralston, A. "De Bruijn Sequences – A Model Example of the Interaction of Discrete Mathematics and Computer Science", *Mathematics Magazine* 55(3):131–143, 1982.

[9] Rose, JA., Deaton, RJ., Franceschetti, DR., Garzon, M. and Stevens, Jr. SE. "A Statistical Mechanical Treatment of Error in the Annealing Biostep of DNA Computation," *Proc. of the Genetic and Evolutionary Computation Conference 1999 Orlando FL (GECCO99),* vol. 2, pp.1829–1834, 1999.

[10] Sänger, W. "Principles of Nucleic Acid Structure", Springer Verlag, 1984.

[11] Sakamoto, K., Kiga, D., Komiya, K., Gouzu, H., Yokoyama, S., Ikeda, S., Sugiyama, H., Hagiya, M. "State Transitions by Molecules", *Biosystems* 52:81–91, 1999.