Exploiting Constraints in the Codeword Design

Filomena de Santis, Nicola Di Luca, Gennaro Iaccarino

***Abstract*—The sequence design is a crucial problem in DNA based computation. We present results about tests carried on sets of DNA strands proposed in DNA computing papers since 1994. Our goal is to direct attention on the necessity of a middle-process between logical designing of input and practical computation in laboratory. The analysis shows that many input sets should not be used for a real DNA computation because they lead to a high probability of incurring in biological faults which result to be very unsafe for a reliable computation.**

***Index Terms*—DNA Computing, *in-vitro* computations, sequence design.**

I. Introduction

Many theoretical models of DNA Computing assume that the computation is errorless. Adleman [1] and Lipton [19-20], for instance, in their experiments used an input constituted by random strands supposing that the probability of errors due to undesired and unexpected behaviors of filaments during computation was negligible. However, it was empirically proved [25] that random sequences are inappropriate for an efficient computation, especially when the input solution size increases. The *codeword design problem*, defined in 2004 by Garzon and Deaton [9] consists in mapping the input instance of a problem in DNA strands that might ensure, with a high reliability level, that chemical reactions such as mismatched hybridization, shift hybridization and hairpin are avoided. Unfortunately, the codeword design problem has been proved to be NP-Complete [9], and, thus, many scientists use evolutionary and probabilistic approaches, or genetics algorithms in order to obtain nearly optimal sequences. In 1998 Deaton et al. used the Hamming distance to avoid hybridization errors. In 2000, Arita et al. [2] developed a design system based on a genetic algorithm applying four specific fitness criteria. In 2001, Tanaka et al. [28] proposed a list of design criterions to be satisfied, and a sequence origination technique based on simulated annealing. In the same year, Ruben et al. [24] build up PUNCH (Princeton University Nucleotide Computing Heuristic), a system for the optimization of DNA input solutions applying genetic algorithms on a simple two-dimensional matrix. In 2002, Tuplan et al. [29] used a stochastic method in order to generate automatically good DNA strands. More recently, Shin et al [14] formulated the DNA sequence design as a multiobjective optimization problem and solved it using a constrained multiobjective evolutionary algorithm. Real-word DNA sequence design is today an open question for in vitro computation, and all kinds of sequence generators only improve input design, without considering whether it complies with the instance of the problem.

In this paper we present a set of tests carried on DNA sequences which were used in the literature as input to molecular algorithms starting from 1994. The aim is to analyze how strong can be the influence of a set of well defined biological constraints on the input choice with respect to the outcome of the computations. The tests were done by means of DNAEdit, a Java application [43] specifically devised for helping in the codeword design problem.

Manuscript received March 22, 2007.

F. de Santis is with ISISLab - Dipartimento di Informatica ed Applicazioni "R.M. Capocelli" - Università degli Studi di Salerno. Via Ponte don Melillo, 84084 Fisciano (SA) - Italy (phone: 0039089969724; fax: 0039089969600; e-mail: fds@dia.unisa.it).

N. Di Luca is with Dipartimento di Informatica ed Applicazioni "R.M. Capocelli" - Università degli Studi di Salerno. Via Ponte don Melillo, 84084 Fisciano (SA) - Italy (e-mail: nicola.diluca@gmail.com).

G. Iaccarino is with ISISLab - Dipartimento di Informatica ed Applicazioni "R.M. Capocelli" - Università degli Studi di Salerno. Via Ponte don Melillo, 84084 Fisciano (SA) - Italy (e-mail: iaccarino@dia.unisa.it).

II. DNA Sequence Design

The DNA computing uses short DNA single strands (oligonucleotides) as memorization and processing unities. The aim of the computing course is simply that of allowing the assembly of single strands in longer DNA molecules by means of the *hybridization* process: the solution to a problem may be seen, indeed, as an extended DNA strand whose chain depends on the input filaments. However, the hybridization process requires that the oligonucleotides combine themselves in a selective mode well-suited to the computation goals. The hybridization between a DNA sequence and its base-pair complement is, indeed, the most important factor to retrieve the information stored in the DNA sequences and operate correctly the computation processes. For this reason, DNA computations need a set of DNA sequences which form stable double strands on one side, and ensure, on the other, that two no complementary sequences do not interact. Non interacting or unstable sequences should be even forbidden to advantage any perfectly matched double strand arising from a DNA sequence and its complement [5]. Thus, partially complementary sequences (mismatched hybridization), sequences matching as result of a shift (shifted hybridization), and sequences interacting with themselves up to form a secondary structure (hairpin) should be avoided, as well as

sequences that do not present uniform chemical attributes.

Since the sequence design is an essential prerequisite for a successful DNA computing, some project *constraints* have been introduced with the aim of *forcing* oligonucleotides to exhibit features which can avoid, or at least reduce, the occurrence of computation errors, such as wrong hybridizations, undesired secondary structures, and inconsistency among sequences. As shown in [14], project constraints can be classified with respect to four *evaluation criterions*: I. preventing undesired reactions; II. controlling secondary structures; III. controlling the chemical attributes of DNA sequences; and IV. restricting DNA sequences. Although a high-quality system should deal with all the previous constraints at the same time, it is worthy noticing that such an implementation is a very difficult task because of the large variety of requirements of each computational architecture.

Before surveying the features of the DNA sequence constraints, let us recall a few of useful definitions.
Let $\sum^*$, with $\sum = \{ A, C, G, T\}$, be the alphabet of the four nucleotide bases making up DNA strands and $\neg x \in \sum$ the complementary base of $x \in \sum$ (A is the complement of T, C is the complement of G and vice versa, as in the Watson-Crick scheme). Then, $X \in \sum^*$, $X=x_1,x_2,\ldots, x_{n-1},x_n$. is a generic DNA sequence of *n* nucleotides, $X^R \in \sum^*$ is the *reverse* of X, and $X^C \in \sum^*$ is the *complement* of X:

$$X^R=x_n,x_{n-1},x_{n-2},\ldots,x_2,x_1 \text{ and } X^C=\neg x_1,\neg x_1,\ldots\neg x_{n-1},\neg x_n,$$

The Watson-Crick (WC) complement of a sequence *X* is :

$$\overline{X} = (X^R)^C$$

III. Preventing undesired reactions

This criterion forces the set of sequences to form the duplexes (i.e. double helix) between a given DNA sequence and its complement, only. It includes a remarkable part of the design constraints: Hamming distance [14], H-measure, Similarity and Complementarity to 3'-end. We briefly describe each of such constraints in the sequel.

A. Hamming Distance

The *Hamming Distance* between two DNA sequences is the number of corresponding places where two bases are complementary [2]. For oligonucleotides *x* and *y* the Hamming distance H(x, y) is given by lining up *x*, the reverse complement of *y*, and subtracting from their common length the number of identical matches. Under appropriate reaction conditions, all the oligonucleotides falling within a certain Hamming distance can hybridize.

B. H-Measure

Let *x* and *y* be two DNA strands of any length. The *H-measure* is defined as follows:

$$|x,y| = \min_{-n<k<n} H\left(x, s^k\left(\overline{y}\right)\right) \qquad (1)$$

where H(*,*) denotes the Hamming distance, σ^k the right (left) shift in case of $k > 0$ ($k < 0$), k the number of shifts, and *y* the complementary pair. The H-measure returns the minimum of all Hamming distances obtained by successively shifting and lining up the WC-complement of *y* against *x*. So a large H-measure indicates a good probability that *y* anneals with *x*, provided appropriate chemical conditions do hold.

C. Similarity

It derives from the H-measure and identifies subsequences with the same structure. To the contrary of H-measure, that compares sequences in opposite directions 3'-5' and 5-3, the similarity compares sequences in the same direction 3'-5' or 5'-3'. In [28], the similarity measure is defined as:

$$Sim = \max_{i,j,i<j} \max_{-n<k<n} \left\{n - H\left(x, s^k\left(x_j\right)\right)\right\} \qquad (2)$$

where x_j is a DNA sequence.

D. 3'-end complementarity

In a sticky-end computation, the 3'-end is the most important constraint for miss-hybridization (i.e. mismatched hybridization). The 3'-end complementarity imposes that only logical complementary strands can hybridize with physical too. A 3'-end miss-hybridization can break the whole computation.

IV. Controlling secondary structures

Secondary structures are usually formed by the interaction of single DNA strands. They include hairpin loops derived from self complementarity and continuity.

A. Self complementarity

When x and y are subsequences of the same DNA strand, self complementarity occurs. This is a crucial fact that directly derives from the H-measure definition, since self complementarity induces a single DNA strand to anneal itself making unfeasible the sequence. Thus, a good design must avoid self complementarity, and each single strand must have the property that any subsequence is not complementary to any other ensuring that the H-measure will be always smaller than the minimal allowed one. Self Complementarity is defined as:

$$Self = \max_{i} \max_{-n<k<n} \left\{n - H\left(x, s^k\left(\overline{x}_i\right)\right)\right\} \qquad (3)$$

B. Continuity

A reaction can not be well controlled if a nucleotide contiguous occurrence in a strand is high: the structure of the sequence, indeed, becomes unstable, and the probability of self complementarity or accidental hybridization increases. Continuity is defined as:

$$Cont = \sum_{i=1}^{m}\sum_{j=1}^{n}(j-1)N_j^i \quad (4)$$

where N^i denotes the number of times the same base appears j-times contiguously in the DNA sequence x_i.

V. Controlling chemical ATTRIBUTES

In many cases, it is desirable to control DNA sequences to have similar chemical characteristics. Measures for this criterion includes GC content and temperature of melting.

A. GC content

GC content is the indicator of the melting temperature. Biological procedures of annealing and melting are just realized by means of a change in the solution temperature. GC di-nucleotide provide a rise of the molecular break temperature allowing the procedure success. DNA strands with largely different GC contents have different melting temperature. Therefore, it is more appropriate to have an identical GC content for all sequences. Moreover, it is better to prefer C with respect to G, since G can form a base-pair with T, even though it is not T-complementary. GC content is defined as:

$$GC_{Content} = \sum_{i=1}^{m}(GC^{(i)} - GC^{(i)}_{user\_defined}) \quad (5)$$

where $GC_{user\_defined}$ is the target value of GC content of sequence x_i. Usually it is chosen as the 50% of strands in solution.

B. Melting Temperature

It is defined as the temperature at which the 50% of the oligonucleotides and their perfect complements couple themselves, whereas the remaining 50% split themselves. Different compositions or sizes of DNA strands can affect the melting temperature in the solution. At the same way of GC content, the Melting Temperature depends on the computation strategy. It is defined as [28]:

$$TM = \sum_{i=1}^{m}(Tm^{(i)} - Tm^{(i)}_{user\_defined}) \quad (6)$$

where $Tm_{user\_defined}$ is the target value of TM for the DNA sequence x_i that biologists can set with respect to the computation.

VI. Restricting DNA sequences

This criterion restricts the composition of a DNA sequence. In some cases a word can be used for special purposes [14]. For example, restriction sites can be used for cut and paste operations on sequences by using nuclease and ligase biological steps. In order to understand what kind of subsequences can be used as special purpose word, some enzyme restriction sites are listed in the Table 1.

VII. Testing DNA sequences

We have tested the use of above described constraints on a variety of DNA sequence sets, which were proposed as input in scientific works in two different period of time: from 1994 to 2000, and from 2000 to nowadays. The year 2000, indeed, marks time when studies on the codeword design problem were born.

Enzyme	Living organism	Restriction site	Nuclease/Ligase
BamHI	Bacillus amyloliquefaciens H	---GGATCC--- ---CCTAGG---	---G 3' 5' GATCC--- ---CCTAG 5' 3' G---
BglII	Bacillus globigi	---AGATCT--- ---TCTAGA---	---A 3' 5' GATCT--- ---TCTAG 5' 3' A---
EcoRI	E.coli RY13	---GAATTC--- ---CTTAAG---	---G 3' 5' AATTC--- ---CTTAA 5' 3' G---
HaeII	Haemophilus aegyptius	---RGCGCY--- ---YCGCGR---	---R 3' 5' GCGCY--- ---YCGCG 5' 3' R---
HindIII	Haemophilus influenzae R	---AAGCTT--- ---TTCGAA---	---A 3' 5' AGCTT--- ---TTCGA 5' 3' A---
PstI	Providencia	---CTGCAG--- ---GACGTC---	---C 3' 5' TGCAG--- ---GACGT 5' 3' C---
SalI	Streptomyces albus G	---GTCGAC--- ---CAGCTG---	---G 3' 5' TCGAC--- ---CAGCT 5' 3' G---
SmaI	Serratia marcescens	---CCCGGG--- ---GGGCCC---	---CCC 3' 5' GGG--- ---GGG 5' 3' CCC---
HaeIII	Haemophilus egyptius	---GGCC--- ---CCGG---	---GG 3' 5' CC--- ---CC 5' 3' GG---
HhaI	Haemophilus hemolyticus	---GCGC--- ---CGCG---	---G 3' 5' CGC--- ---CGC5' 3' G---
HpaII	Haemophilus parainfluenzae	---CCGG--- ---GGCC---	---C 3' 5' CGG--- ---GGC 5' 3' C---
Sau3A	Staphylococcus aureus 3A	---GATC--- ---CTAG---	--- 3' 5' GATC--- ---CTAG 5' 3' ---
NotI	Nocardia otitidis-caviarum	---GCGGCCGC--- ---CGCCGGCG---	---GC 3' 5' GGCCGC-- ---CGCCGGC 5' 3' G--

Table 1. Restriction sites of the most used enzymes.

In order to accomplish the test, we have used DNAEdit, that is to say a Java application specifically devised and implemented with the aim to help scientists to choose a good set a codewords. DNAEdit makes use of string matching algorithms to verify whether or not a given set of input DNA sequences comply with any of the constrains, and returns a report containing information about compliance, as well as error probability which it is possible to fall into during real computation.

In the two following tables we show the results obtained by using DNAEdit. The first row of each table contains the references to works where sequences were taken on, whereas the first column contains the constraints we looked upon. For each DNA input set we marked in black the constraints which

were not complied with.

Table II shows results of our experiments on sets of DNA strands belonging to the period 1994 - 2000. Table III shows results of our experiments on sets of DNA strands belonging to the period next to 2000. They clearly show that, after 2000, the codeword choice has been more accurate in compliance with constraints and the error probability decrease in computation.

	[4]	[19]	[10]	[11]	[13]	[31]	[18]	[22]	[23]	[30]
Hamming D.	•		•		•	•		•		•
H-measure		•		•			•		•	•
Similarity	•	•			•	•		•		
3'-end C.		•	•	•			•		•	
Self comp.	•					•		•		•
Continuity	•		•		•		•		•	•
GC Content		•		•	•		•	•		•
TM	•		•			•			•	

Table II. Constraints evaluation on DNA strands from 1994 to 2000 .

	[21]	[3]	[15]	[27]	[26]	[17]	[12]	[6]	[7]	[16]
Hamming D.	•					•				
H-measure				•				•		•
Similarity	•		•			•			•	
3'-end C.	•	•					•			
Self comp.				•						•
Continuity			•		•		•	•		
GC Content						•			•	
TM		•								•

Table III. Constraints evaluation on DNA strands after 2000 .

Figure 1 shows the percentage of computational errors for each constraints in works pre and post 2000; it also shows a remarkable difference among physical constraints such as Hamming distance, H-measure etc., and the chemical ones such as GC content, MT, etc.

VIII. Conclusions

In this paper we present results concerning tests about biological constraints on sets of DNA sequences involved in *in-vitro* computations since 1994. Tests show that many experiments should be executed in laboratory before choosing a data set as input, because the probability of biological fault is very high, and, consequently, the probability of erroneous computations is very high, too.

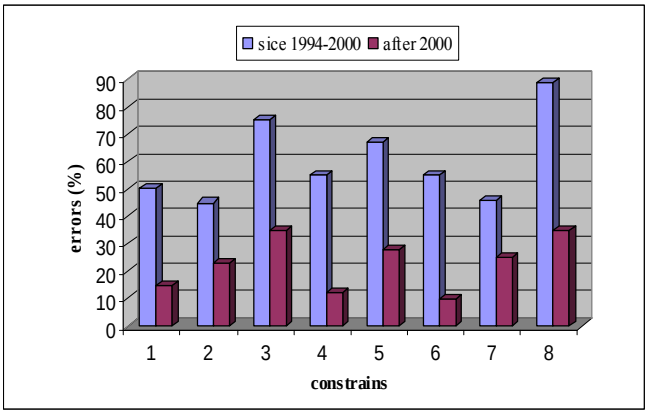

Figure 1. A graphical representation of biological faults in DNA computations sice 1994. The constrains order is: 1)Hamming Distance, 2) H-measure, 3) Similarity 4)3'-end Complementarity, 5)Self Complementarity, 6) Continuity, 7)GC Content, 8) Melting Temperature.

Our goal was that of underlining the importance of the biological constraints when the input is chosen for DNA computations, and the influence they have in the step of design and implementation of molecular algorithms.

Eventually, we longed for remarking the value of support tools, such as DNAEdit, for the help in the design of molecular algorithms: they can assume, indeed, the role of "middle-man" instruments between the theoretical realization and the practical laboratory implementation for any DNA computing procedure.

References

[1] L. Adleman, "Molecular Computation of Solutions to Combinatorial Problems". *Science* vol. 266: pp 1021-1024, Nov. 11, 1994.

[2] M. Arita, A. Nishikawa, M. Hagiya, K. Komiya, H. Gouzu, K. Sakamoto,,"Improving Sequence Design for DNA Computing," in *Proceedings of Genetic and Evolutionary Computation Conference 2000*, pp. 875-882, 2000.

[3] R.Barua, J. Misra, "Binary Arithmetic for DNA Computers". *Lecture Notes In Computer Science*; Vol. 2568, pp. 124-132, 2002.

[4] D. Boneh, C. Dunworth, R. J. Lipton. "Breaking DES Using a Molecular Computer". Technical Report CS-TR-489-95. Department of Computer Science, Princeton University, 1995.

[5] A. Brenneman and A. Condon, "Strand design for biomolecular computation," *Theor. Comput. Sci.*, vol. 287, pp. 39–58, 2002.

[6] W.L. Chang, M. Guo, "Fast Parallel Molecular Algoriths for DNA-Based Computation: Factoring Integers". *IEE Transactions on Nanobioscience*, vol. 4, n. 2, June 2005.

[7] J. Chen et al. "A DNA-based Memory with in-vitro learning and associative recall". Natural Computing (4), pp. 83-101, 2005.

[8] R. Deaton, M. Garzon, R. Murphy, J. A. Rose, D. R. Franceschetti, S. E. Stevens, "Reliability and Efficience of a DNA-based Computation". *Physical Review Letters*, vol. 80, no. 2, pp. 417–420, 1998.

[9] M. Garzon, R. Deaton. "Codeword design and information encoding in DNA ensembles". *Natural Computing*, vol. 3, pp. 253-292. 2004

[10] F. Guarnieri, M. Fliss, C. Bancroft. "Making DNA Add", *Science*, vol. 273, pp. 220-223, July 1996.

[11] M. Hagiya, "Perspectives on molecular computing" *New Generation Computing*. 17(2), 131-140, 1999.

[12] Z. Ibrahim, Y. Tsuboi, 0. Ono, M. Khalid, "Molecular Computation Approach to Compete Dijakstra's Algorith". *In Proc. of 5th Asian Control Conference*, 2005.

[13] P. Janczak, T. Mulawka, J.J. Plucienniczak, A "Inference Via DNA Computing", *Proc. Congress on Evolutionary. Computation* (CEC'99), 2, Washington, USA, pp. 988-393, 1999.

[14] D. Kim, S.-Y. Shin, I.-H. Lee, , B.-T. Zhang, "Multi-objective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing," IEEE Transaction on Evolutionary Computation, vol. 9, n.2, April 2005.

[15] C. Lampasona. "DNA Computers Applications: Cryptography". Innovative Computer Architectures and Concepts, June 2002.

[16] H. Lederman, Joanne Macdonald, Darko Stefanovic, and Milan N. Stojanovic, "Deoxyribozyme-Based Three-Input Logic Gates and Construction of a Molecular". *Biochemistry* (45), pp.1194-1199, 2006.

[17] J.Y. Lee et al. "Solving Traveling Salesman Problems with DNA Molecules Encoding Numerical Values", *Biosystems* (78), pp. 39-47, 2004.

[18] A. Leier, C. Richter, W. Banzhaf, H. Rauhe. "Cryptography with DNA binary strands". *Biosystems* , 2000.

[19] [lip2]R. Lipton. "Using DNA to solve NP Complete Problems". *Science*, vol. 268, pp. 542-545, April 1995.

[20] R. Lipton. "DNA Solution of Hard Computational Problems", *Science*, vol. 268, pp. 542-545, April 1995.

[21] M. Mehta. "Evalution of Two Alternative Solutions for Improving Computer Security". Technical Report, August 2001.

[22] H. Rauhe, G. Vopper, U. Feldkamp, W. Banzhaf, J. C. Howard. "Digital DNA Molecules". *Proceedings 6th DIMACS Workshop on DNA Based Computers*, Leiden, The Netherlands, 13 - 17 June 2000.

[23] C. Richter, A. Leier, W. Banzhaf, H. Rauhe, "Private and Public Key DNA steganography". *Proceedings 6th DIMACS Workshop on DNA Based Computers*, Leiden, The Netherlands, 13 - 17 June 2000.

[24] A. J. Ruben, S. J. Freeland, L. Landweber, "PUNCH: an Evolutionary Algorithm for Optimizing Bit Set Selection," in *Proceedings of the 7th International Workshop on DNA-Based Computers*, pp. 260-270, 2001.

[25] J. Sager, D. Stefanovic. "Designing Nucleotide Sequences for Computation: A Survey of Constraints". DNA computing: 11th International Workshop on DNA Computing, pp. 275-289, 2005.

[26] G. Smith et al. "Some Possible Codes for Encrypting Data in DNA". *Biotechnology Letters* (25), pp. 1125-1130, 2003.

[27] M. N. Stojanovic, D. Stefanovic, "Deoxyribozyme-Based Half-Adder". *JACS* 2002.

[28] F. Tanaka, M. Nakatsugawa, M. Yamamoto, T. Shiba, A. Ohuchi, "Developing Support System for Sequence Design in DNA Computing," in *Proceedings of the 7th International Workshop on DNA-Based Computers*, pp. 340-349, 2001.

[29] D. C. Tuplan, H. Hoose, A. Condon, "Stochastic Local Search Algorithm for DNA Word Design," in *Proceedings of 8th International Workshop on DNA-Based Computers*, pp. 229-241, 2002.

[30] P. Wasiewicz, R. Rudnicki, J.J. Mulawka, B. Lesyng. "Adding Numbers with DNA". In *Proceedings 2000 IEEE International Conference on Systems, Man & Cybernetics - SMC2000*, Nashville, USA, 265-270.

[31] P. Wąsiewicz et al. "Implementation of Data Flow Logical Operations via Self–Assembly of DNA". *Lecture Notes in Computer Science* 1586, 1999, Springer, pp. 174–182.

[32] R. B. Wallace et al. "Hybridization of synthetic oligodeoxyribonucleotides to phi chi 174 DNA : The effect of single base pair mismatch". Nucleic Acid Research, vol. 6, n. 11, pp. 3543-3557, 1979.

[33] Weiss R., Basu S. The Device Physics of Cellular Logic Gates, First Workshop on Non-Silicon Computing, Cambridge, 2002, Mass.

[34] J. G. Wetmur. "DNA probes: Applications of the principles of nucleic acid hybridization". Critical Rev.Biochem. Molecular Bio., vol. 26 pp. 227-259, 1991.

[35] J. Santa Lucia. "A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics". In Proc. of National Acad. Sci. U.S.A, vol. 95, pp. 1460-1465, 1998.

[36] B. Wang et al. "A framework for Modeling DNA Based Molecular Systems". Lecture Notes in Computer Science n. 4287, pp. 250-256.

[37] S. Yaegashi et al. "Experimental Validation of the Statistical Thermodynamic Model for Prediction of the Behaviour of Autonomous Molecular Computers Based on DNA Hairpin Formation". Lecture Notes in Computer Science n. 4287, pp. 428-438, 2006.

[38] M. Yamamoto et al. "Aqueous Computing with DNA Hairpin-Based RAM". 10th International Workshop on DNA Computing 2005. Milan, June 7-10, 2005. LNCS n. 3384, pp. 355-364, 2005.

[39] M. Yamamoto et al. "Conformational Addressing Using the Hairpin Structure of Single-Strand DNA". *9th International Workshop on DNA Based Computers, DNA9,* Madison, WI, USA, June 1-3, 2003,

[40] M. Yamamoto et al. "Sequence Design for Stable DNA Tiles". Lecture Notes in Computer Science (DNA 2006), n.4287, pp. 172-181, 2006.

[41] M. Yamamoto et al. "Unravel Four Hairpins!". Lecture Notes in Computer Science (DNA06), n. 4287, pp. 381-392, 2006.

[42] X. Zhu, W. Liu. "Template Frame for DNA Computing". Poster Proc. of The 12th International Meeting on DNA Computing 2006 (DNA12). June 5-9, 2006, Seoul – Korea.

[43] F. de Santis, N. Di Luca, G. Iaccarino. "DNA*Edit*: a Java Tool for the Codeword Problem". Submitted to 2007 Summer Computer Simulation Conference (SCSC'07). July 15-18, 2007. San Diego, CA, USA.