A Conventional Strands Evaluator for DNA Computations

Filomena de Santis
DIA - Università degli Studi di Salerno
Via Ponte don Melillo,84084 Fisciano (SA), Italy
+39 089 96 9724
fds@dia.unisa.it

Gennaro Iaccarino
DIA - Università degli Studi di Salerno
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
+39 089 96 9313
iaccarino@dia.unisa.it

ABSTRACT
The DNA sequence design is a crucial problem in DNA based computations. In the literature there is evidence that many input sets should not be used for real DNA computations. This approach might lead to high probabilities of incurring in biological faults which make computations unsafe. In this paper we present an intermediate tool between strand logical design and practical computations, that allows scientists to approach in-vitro computations reducing the probability of biological mistakes in the phase of theoretical input design for practical computations.

Keywords
DNA Computing, *in-vitro* computations, codeword design problem.

1. INTRODUCTION
Many theoretical models of DNA computing assume that the computation is errorless. Adleman [1] and Lipton [20-21], for instance, in their experiments used as input 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 [26] 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 [10], 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 belong to NP-Complete class [10], and, thus, many scientists use evolutionary and probabilistic approaches, or genetics algorithms in order to obtain nearly optimal sequences [2, 15, 25, 29, 30].
In this paper we present a simple tool specifically devised for helping in the input sequences design; this application allows users to create, modify, visualize, evaluate, and store a pool of input sequences complying with a set of well defined *project constraints*, and obtain stepwise reports about their effectiveness for real biological computations (in laboratory). Differently from already known applications [15, 25, 29], it is not a sequence generator, that doesn't respect the specification of the problem to handle, but an auxiliary process that improve the input sequence design.

BIONETICS '07 , December 10–13, 2007, Budapest, Hungary.

2. BIOLOGICAL CONSTRAINS
The DNA computing uses short DNA single strands (oligonucleotides) as memorization and processing unities. The aim of the computing path 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 is, in fact, 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 activates 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; perfectly matched double strands should be, on the contrary, brought about [5]. Namely, partially complementary sequences (mismatched hybridization), sequences matching as a result of shifts (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 successful DNA computations, 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 [9] and in [15], project constraints can be classified with respect to four evaluation criterions.

1) 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 the following measures: the Hamming distance, defined as number of corresponding places where two bases are complementary; the H-measure, defined as the minimum of all Hamming distances obtained by successively shifting and lining up two tested sequences; the Similarity, defined as the inverse of the H-measure between two given DNA sequences (the H-measure compares sequences in opposite directions 3'-5' and 5-3, and the similarity compares sequences in the same direction 3'-5' or 5'-3'.); the 3'-end complementarity that, in some cases, imposes complementary 3'-ends words for hybridization (in a sticky-end computation, a 3'-end miss-hybridization can break the whole computation).

2) *Controlling secondary structures*: secondary structures are usually formed by bad interactions of single DNA strands. They include internal loops and hairpin loops, deriving from bad evaluation of self complementarity and continuity criterions. Self complementarity occurs when complementary subsequences compose the same strand; 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. Continuity occurs when a nucleotide contiguous occurrence in a strand is high: the structure of the sequence, indeed, becomes unstable, and the probability of loops or accidental hybridization increases.

3) *Controlling chemical attributes*: in many cases, it is desirable to control DNA sequences to have similar chemical characteristics. Measures for this criterion include GC-content and melting temperature. GC-content is the percentage of guanine and cytosine in a whole DNA strand; increasing this value it provides a rise of the energy allowed to realize annealing and melting procedures. So, sequences with different GC-content necessitate of different energy for biological steps, altering the computation. Melting temperature 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 temperature in the solution.

4) *Restricting DNA sequences*: this criterion restricts the composition of a DNA sequence. In some cases a list of oligonucleotides can be used for special purposes. Also, special DNA subsequence, such as restriction enzyme sites, should be controlled for proper reactions of nuclease or ligase [15].

3. DNA*Edit* Tool

DNA*Edit* is the acronym of Deoxyribose Nucleic Acid Editor, a sequence design system useful as a middle-process tool between logical sequences design and biological implementation. The aim of DNA*Edit* is two-fold: creating *ex novo* a library of DNA sequences, and evaluating input sequences with respect to the above mentioned constraints. As shown in [29], and widely discussed in other works, a sequence design system must satisfy at least two collections of requisites. First of all, users must be able to define the sequence sizes without any limit of length, to select the design constraints to be exploited in the evaluation process, to point out sequence positions to be spent as possible restriction sites, and eventually to examine the righteousness of the obtained sequences in sight of a probable reuse of the system. Second, the design system must guarantee reliability, analysis resources, and reuse of sequences, as well as a very friendly interface.

Many design systems have been implemented since the codeword design problem was defined. NACST (*Nucleic Acid Computing Simulation Toolkit*) [15] is among the best known ones. By means of a genetic algorithm, it generates a fixed number of pools of sequences optimizing a multiobjective fitness function. However, it massively uses a pre-existent set of sequences, restricts a pool to have the same length for all sequences, does not allow to recognize the library elements which could be invalidated by sequences non complying with the constraint thresholds, and requires users that are very familiar with evolutionary approaches to optimization.

DNA*Edit* provides a different approach to the sequence design problem: the user, indeed, absolutely arbitrates the decision about the sequences he wants to work with, creates his own library, and, thereafter, proceeds with the sequence evaluations establishing whether the design has been satisfactory or needs a further elaboration in order to improve the sequence quality. That is to say, the sequence generation does not depends on the system, but exclusively on the user. Moreover, no limits at all are imposed on the length of the sequence pools, no uncertainty is introduced on possibly mismatching sequences since the effective value of each constraints is clearly given in the final report, as well as a comparison of values (i.e. H-measure, similarity, etc.), when necessary, no a priori technical knowledge is required to the user apart from those concerning the sequence design.

3.1 Input procedure

The user can introduce a new set of sequences into the system or enlarge an old one, either by simulating the artificial synthesis of DNA or making use of facilities that allow to have a complete list of sequences recognized by restriction enzymes, add new sequences at the end of the existing ones, specify a high number of sequences to be repeated almost similarly, and make available classical operations of cut, copy and paste. Obviously, the library itself can be fully visualized and modified, once it has been build up. During this initial procedure sequences are scanned, in run time, in order to point out substrings representing restriction sites or special purpose words, in accordance with the fourth above mentioned constraint class.

3.2 Constraints evaluation

After the sequence library creation, the user can decide to evaluate its righteousness with respect to one or more design constraints. Formally, the DNA sequence design problem can be written as follows [15]:

$$\textit{Minimize } F(x)=(f_1(x), f_2(x),\ldots, f_n(x));$$
$$f_i(x)\in\{\textit{Biological Constraints}\}$$

Our goal is to point out constraints that prevent the minimization of *F(x)*. In the following, the behaviour of our system beside biological constraints is described.

3.2.1 Preventing Undesired Reactions

DNA*Edit* is designed in order to test Hamming Distance, H-measure, Similarity, and 3'-end Complementarity.

Hamming Distance procedure: let x and y be two no complementary sequences, for example x=5'-AGGCTTTAGC-3' and y=5'-CGAAATCGAA-3'. The Hamming distance is computed by lining up x and the reverse complement of y (WC complement) and subtracting the positions in which bases are the same:

5'-AGGCT**TT**AGC-3'
3'-TTCGA**TT**TCG-5'

In our case, it results that H(x, y) = 2. In a good computation, each pair of no complementary sequences should have a minimal value of H(x, y).

H-Measure evaluation: The H-measure is calculated by starting from Hamming distance and shifting x against y for all the possible positions. So a large H-measure indicates a good probability that y anneals with x. Our system illustrates the Hamming distance and the H-measure for each pair of input sequences at the same time. The H-measure is defined formally

below:

$$|x, y| = \min_{-n<k<n} H\left(x, \sigma^k\left(\bar{y}\right)\right) \quad (1)$$

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.

Similarity check: The similarity check procedure is implemented at the same of H-measure check procedure, apart from the fact that sequences are not compared in the reverse mode, but in the same direction (3'-5' or 5'-3'). Results are reported in a windows outline.

3'-end complementarity check: This procedure, directly deriving from the Hamming distance evaluation, checks all the input sequences that have 3'-end complementary extremities. The goal of this procedure is to indicate whether 3'-end complementarity exists between sequences, and the user is allowed to decide whether or not it is convenient to modify input strands according to his kind of computation. The 3'-end complementarity check procedure is very efficient in sticky-end and on-surfaces computations.

3.2.2 Secondary Structures

DNA*Edit* allows to evaluate the presence of secondary structures in the input sequences; as already mentioned, the secondary structures might strongly threat the computation success. Once again, the report is shown to the user leaving to him the faculty to change the input sequence in accordance with the computation.

Self Complementarity evaluation: It is the main cause of hairpin in DNA structures. It depends on sections of sequences that are complementary among themselves. Self complementary evaluation procedure determines the parts of a sequence that are complementary themselves and the distance between them. Obviously, the probability of hairpins or the deterioration of the sequence increases as the distance increases. The formal definition of the procedure is the following:

$$Self = \max_i \max_{-n<k<n} \left\{ n - H\left(x, \sigma^k\left(\bar{x}_i\right)\right) \right\} \quad (2)$$

Continuity evaluation: if the same bases occur many times in a DNA strand, the sequence can show an unexpected structure. DNAEdit simply evaluates continuity by counting substring formed by the same base (A, C, G, T), and reporting it. below the used procedure for continuity is proposed:

$$Cont = \sum_{i=1}^{m} \sum_{j=1}^{n} (j-1) N_j^i \quad (3)$$

where N^i denotes the number of times the same base appears j-times contiguously in the DNA sequence x_i. Figure 1 shows a report of this procedure.

3.2.3 Chemical Attributes

DNA*Edit* calculates, for each set of input sequences, the *GC* content ant the melting temperature.

GC content evaluation: It is one of the main causes of fault in DNA computations. DNA*Edit* simply calculates the *GC* content by scanning each input sequence and counting the number of CG bases. The procedure returns this value in percentage. Figure 1 shows the report for the GC content procedure. GC content is defined as:

$$GC_{Content} = \sum_{i=1}^{m} (GC^{(i)} - GC^{(i)}_{user\_defined}) \quad (4)$$

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.

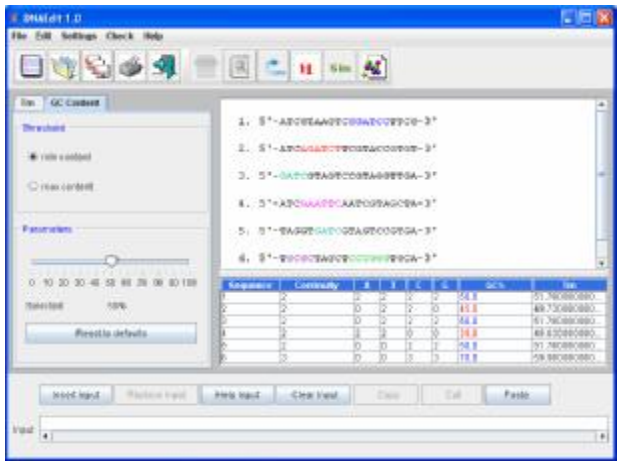

Figure 1. Continuity and GC evaluation. Continuity is expressed for each base in each sequence. Report shows the number of adjacencies for each base and the greats is assigned as "continuity". GC content is expressed in percentage; the user can choose (minimal or maximal) limit values in according with his computation.

Melting Temperature evaluation: the Melting Temperature (MT) is one of the most important features for laboratory experiments. There are many theoretical methods to calculate the melting temperature, and we have chosen three different methods: GC ratio [33], Salt Adjusted [32] and Nearest Neighbour model [34]. In the GC ratio model, the MT depends only on the percentage of G and C bases in the sequences. In the Salt Adjusted it depends on the salt concentration in the solution. In the Nearest Neighbour model it depends on the salt concentration in the solution and on the quantity of DNA. The Melting Temperature check process allows to choose among these three methods and gives default values for each method. General procedure is defined as follows:

$$TM = \sum_{i=1}^{m} (Tm^{(i)} - Tm^{(i)}_{user\_defined}) \quad (5)$$

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 [29]. Figure 2 shows a report of the Basic method.

4. EXPERIMENTAL RESULTS

DNA*Edit* was used to test a variety of DNA sequences, which were proposed as input of DNA computations in scientific works from 1994 to 2006. The sample of sequences analyzed refers to papers 3, 4, 6, 7, 11, 12, 13, 14, 16, 17, 18, 19, 20, 23, 24, 27, 28, 31. Results shows that many of the pool set used for theoretical computations can't be used for real procedures in laboratory, because biological fault should incur with more probability. Bar chart, in figure 3, shows the probability of biological faults, in works pre and post 2000. It is evident that input sets, developed after the year 2000 (time when studies on the codeword design problem were born), are more streamlined than the input sets

designed at the end of the past century. It is also interesting to notice the remarkable difference among physical constraints such as Hamming distance, H-measure etc., and chemical ones such as GC content, MT, etc. Figure 4 shows the trend of sequences that do not satisfy the constraints imposed by the four criterions of section 2. It is important to notice that most structural constraints decrease until about the 20%, whereas for chemical constraints (CG content e Melting Temperature) the percentage is still about 40% . The results of tests clearly show the importance of a design framework, such as DNA*Edit,* able to evaluate in real time the features of the input pool that the user is going to generate in such a way that the biological computation turns to be out correctly.

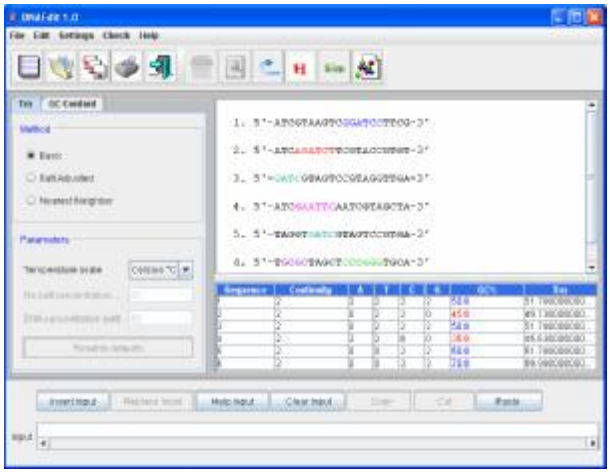

Figure 2. MT evaluation. The report shows the melting temperature in three different unit of measurement (C, K, F). The process returns the melting temperature by using three different methods. Temperatures out of limits are signed with red color.

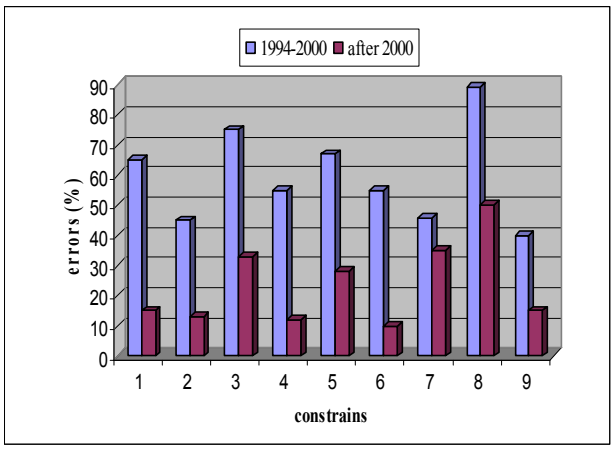

Figure 3. Graphical representation of biological faults in DNA computations since 1994. The constraints order is: 1) Hamming Distance, 2) H-measure, 3) Similarity 4) 3'-end Complementarity, 5) Self Complementarity, 6) Continuity, 7)GC Content, 8) Melting Temperature, 9) Use of restriction sites. The sets of DNA input strands have been taken from DNA computing works since 1994 to 2006.

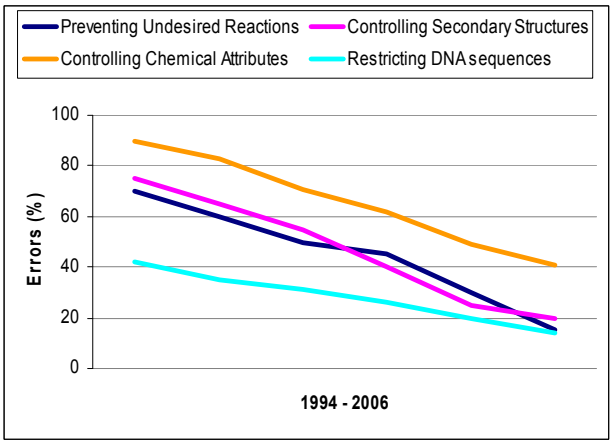

Figure 4. Graphical representation of the error performance due to a missed management of input constraints. Constraints have been grouped in the four classes related in section 2. The data sample tested with DNA*Edit* refers to papers since 1994 to 2006.

5. CONCLUSIONS

The DNA sequence design problem is actually an open problem and many scientists aim to resolve it by implementing sequences generators or compilers. DNA*Edit* is a simple application which takes in input DNA sequences as strings and returns a report in which biological constraints are evaluated and possible mistaken computations are highlighted. The goal is not replace the input designer, but simply helps him in modeling sequences that resolve the primary computer science problem, minimizing the probability of mistakes during the computation. We have also seen that in the last years, scientists have designed input strands more compliant with biological constraints while resolving problems with in-vitro computations. On this assumption, we consider this kind of applications as an important step towards practical DNA computation without faults.

6. REFERENCES

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

[2] M. Arita, et al. "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 489-95, 1995.

[5] A. Brenneman and A. Condon, "Strand design for biomolecular computation," Theor. Comp. Scien., 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] F. de Santis, N. Di Luca, G. Iaccarino. “Exploiting Constraints in the Codeword Design”. *Lecture Notes in Engineering and Comp. Science,* v. II, pp. 1455-1459, 2007.

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

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

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

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

[14] P. Janczak, T. Mulawka, J.J. Pluciennicz ak, A “Inference Via DNA Computing”, *Con. on Evolutionary. Computation* (CEC’99) Washington, USA, pp. 988-393, 1999.

[15] 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.

[16] C. Lampasona. “DNA Computers Applications: Cryptography”. Innovative Computer Architectures and Concepts, June 2002.

[17] 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.

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

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

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

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

[22] M. Mehta. “Evalution of Two Alternative Solutions for Improving Computer Security”. Technical Report, 2001.

[23] 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.

[24] 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.

[25] A. J. Ruben, S. J. Freeland, L. Landweber, “PUNCH: an Evolutionary Algorithm for Optimizing Bit Set Selection,” *7th International Workshop on DNA-Based Computers*, pp. 260-270, 2001.

[26] J. Sager, D. Stefanovic. “Designing Nucleotide Sequences for Computation: A Survey of Constraints”. *11th Int. Workshop on DNA Computing,* pp. 275-289, 2005.

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

[28] M. N. Stojanovic, D. Stefanovic, “Deoxyribozyme-Based Half-Adder”. *JACS* 2002.

[29] F. Tanaka, et al. “Developing Support System for Sequence Design in DNA Computing,” *7th International Workshop on DNA-Based Computers*, pp. 340-349, 2001.

[30] 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.

[31] P. Wasiewicz, R. Rudnicki, J.J. Mulawka, B. Lesyng. “Adding Numbers with DNA”, *Proceedings of IEEE International Conference on Systems, Man & Cybernetics*, Nashville, USA, 2000, pp.265-270.

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

[34] J. Santa Lucia. “A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics”. *National Acad. Scien. U.S.A,* vol. 95, pp. 1460-1465, 1998.

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

[36] 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.

[37] A. Yamamoto et al. “Sequence Design for Stable DNA Tiles”. *Lecture Notes in Computer Science (DNA 2006),* n.4287, pp. 172-181, 2006.

[38] M. Yamashita et al. “A probabilistic Model of the DNA Conformational Change”. *2nd International Meeting on DNA Based Computations 2006*. pp. 274-285, 2006.