Scientific Journal Impact Factor (SJIF): 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

DNA Computing and Its Applications and Implementation

Praveen Kr. Rai[1], Shashi Bhushan[2]
[1,2]Assistant Professor, Computer Science & Engineering, IIMT College of Engineering

Abstract—The aim of this manuscript is to illustrate the current state of the art of DNA computing achievements, especially of new approaches or methods contributing to solve either theoretical or application problems. Starting with the NP-problem that Adleman solved by means of wet DNA experiment in 1994, DNA becomes one of appropriate alternatives to overcome the silicon computer limitation. Today, many researchers all over the world concentrate on subjects either to improve available methods used in DNA computing or to suggest a new way to solve engineering or application problems with a DNA computing approach. This paper gives an overview of research achievements in DNA computing and touches on the achievements of improved methods employed in DNA computing as well as in solving application problems. At the end of discussion we address several challenges that DNA computing faces in the society.

I. INTRODUCTION

DNA computing is the research area that is improving fast since DNA molecules are implemented in a computational process. The main objectives of this research area is to produce, in near future, a biologically inspired computer based on DNA molecules to replace or at least beneficially complement with a silicon based computer.

DNA is a basic storage medium for all living cells. The main function of DNA is to absorb and transmit the data of life for billions years. Roughly, it is around 10 trillions of DNA molecules could fit into a space, the size of a marbles. Since all these molecules can process data simultaneously, theoretically, we can calculate 10 trillions times simultaneously in a small space at one time. DNA computing is more generally known as molecular computing. It is interdisciplinary field where it is a combination of biology, chemistry, mathematics and computer science. Computing with DNA offers a completely new paradigm for computation. The main idea of computing with DNA is to encode data in a DNA strand form, and laboratory techniques of molecule biology, called as bio-operations, will be involved to manipulate DNA strands in a test tube in order to simulate arithmetical and logical operations. It is estimated that a mix of 1018 DNA strands could operate 104 times faster than the speed of a today’s advanced supercomputer.

The search for new methods of computing is something that has engaged humankind for as long as history has been recorded. From the abacus and Napier’s bones (1658) to the modern electronic supercomputer, people have continuously sought new ways to automate the task of performing computations.

In recent decades the word ‘‘computer’’ has become synonymous with an electronic computing machine due to the overwhelming success of this particular paradigm. This, however, is only part of the story. Indeed, as we build faster and faster electronic computers, we are beginning to reach physical limits, beyond which our current technology cannot venture.

The first paper to explicitly use DNA as a computational medium was published by Leonard Adleman in 1994 (Adleman, 1994). In this paper, Adleman gave a description of how a combination of DNA, the correct enzymes, and appropriate laboratory protocols can solve an instance of the Traveling Salesman Problem (TSP). What was particularly attractive about this paper was that, in addition to providing the theoretical framework for such a computation, it also provided concrete results from a complete experimental implementation of the proposed protocol. Consequently, it is Adleman's experiment that is generally viewed as the landmark for the genesis of the field of biomolecular computing.

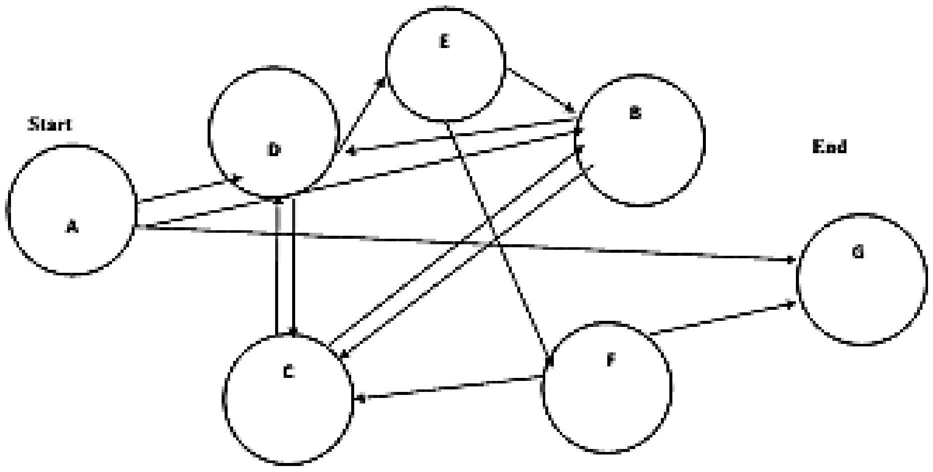
This single DNA computing experiment sparked the interest of a number of researchers in both computer science and molecular biology, and soon there were several competing models of DNA computing. Adleman's choice to solve a problem that is known to be NP-complete put exceedingly high expectations on DNA computing and resulted in some constructive criticism in the form a brief complexity analysis by Hartmanis (1995). In this paper, Hartmanis shows that using Adleman's protocol for solving a 200-node instance of the TSP would require 24 Earth masses of DNA. While this is clearly not feasible, subsequent work in Suyama et al. (1997) showed that modifications of the protocol lead to realistically solving much larger TSP instances. The objection that we raise to Hartmanis's critique is that he evaluated the efficiency of an experiment that was intended as a proof of concept, and an astonishing one for that matter. Similar criticisms of early transistor-based electronic computers would have led us to believe that they too would never become a viable technology.

In this article we introduce and provide a high-level view of a number of unique models and implementations of DNA computing. The results considered have been chosen to reflect a representative sample of key research areas within the field.

The article begins with two introductory sections on basic molecular biology and computer science. We then continue with an exposition of splicing systems and other theoretical models. The fifth section is devoted to more practical, implementation-based approaches. The sixth section provides a brief overview of the new field of in vivo computation, and the final section presents concluding remarks.

II. WHAT IS DNA?

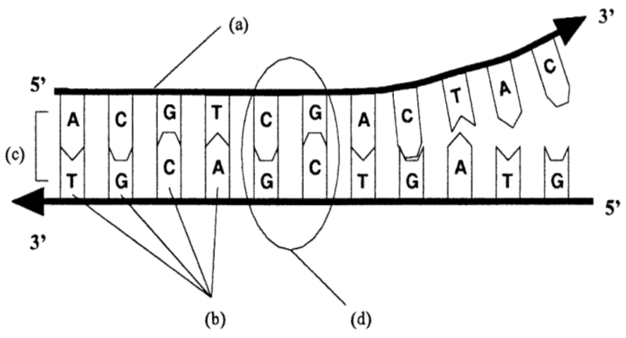
Before delving into the principles of DNA computing, we must have a basic understanding of what DNA actually is. All organisms on this planet are made of the same type of genetic blueprint which bind us together. The way in which that blueprint is coded is the deciding factor as to whether you will be bald, have a bulbous nose, male, female or even whether you will be a human or an oak tree.
Within the cells of any organism is a substance called Deoxyribonucleic Acid (DNA)



This type of problem is known as a non-deterministic polynomial time problem (NP).The idea of guessing the right answer to a problem, or checking all possible problems in parallel to determine which is correct, is called nondeterminism. An algorithm that works in this manner is called a nondeterministic algorithm, and any problem with an algorithm that runs on a non-deterministic machine in polynomial time is called a non-deterministic polynomial time problem.

The NP problem was chosen for Adleman’s DNA computing test as it is a type of problem that is difficult for conventional computers to solve. Conventional computers Adleman, using a basic 7 city, 13 street model for the Traveling Salesman Problem,are better suited for deterministic computation permitting at most one next move at any step in a computation. The inherent parallel computing ability of DNA combinationhowever is perfectly suited for NP problem solving. created randomly sequenced DNA strands 20 bases long to chemically represent each city and a complementary 20 base strand that overlaps each city’s strand halfway to represent each street (Fig.). This representation allowed each multi-city tour to become a piece of double stranded DNA with the cities linked in some order by the streets.

III. BIOLOGICAL CONCEPT FOR DNA COMPUTING



DNA computing is based on the idea that molecular biology processes can be used to perform arithmetic and logic operations on information encoded as DNA strands.

A DNA single strand consists of four different types of units called bases (Figure 1b) strung together by an oriented backbone (Figure 1a) like beads on a wire. The bases are adenine (A), guanine (G), cytosine (C), and thymine (T), and A can chemically bind to an opposing T on another single strand (Figure 1c), while C can similarly bind to G (Figure 1d). Bases that can thus bind are called Watson=Crick (W=C) complementary, and two DNA single strands with opposite orientation and with W=C-complementary bases at each position can bind to each other to form a DNA double strand (Figure 1) in a process called base pairing.

To encode information using DNA, one can choose an encoding scheme mapping the original alphabet onto strings over {A, C, G, T}, and proceed to synthesize the obtained information-encoding strings as DNA single strands. A computation will consists of a succession of bio-operations (Kari, 1997), such as cutting and pasting DNA strands, separating DNA sequences by length, extracting DNA sequences containing a given pattern, or making copies of DNA strands. The DNA strands representing the output of the computation can then be read out and decoded.

Herein lie a wealth of problems to be explored, stemming from the fact that encoding information in DNA differs from encoding it electronically, and bio-operations differ from electronic computer operations. A funda-mental difference arises from the fact that in electronic computing data interaction is fully controlled, while in a test-tube DNA computer, free-floating data-encoding DNA single strands can bind because of W=C complementarity. Another difference is that in DNA computing, a bio-operation usually consumes both operands. This implies that if one of the operands is either involved in an illegal binding or has been consumed by a previous bio-operation, it is unavailable for the desired computation. Yet another difference is that while in electronic computing a bit is a single individual element, in DNA experiments each submicroscopic DNA molecule is usually present in millions of identical copies. The bio-operations operate in a massively parallel fashion on all identical strands. However, this process is governed by the laws of chemistry and thermodynamics, and the output obeys statistical laws.

Differences like the ones just mentioned point to the fact that a new approach should be employed when analyzing and processing DNA-encoded information. These differences are starting to be tapped into by research in DNA computing, as will become apparent in the examples presented in this article. For more molecular biology terminology and notions the reader is referred to Kari (1997), Watson et al. (1987), and Calladin and Drew (1999).

IV. RESEARCH AND ENGINEERING

DNA Chemistry is totally different for DNA Mathematical calculation. It may take time. May be executed in minutes or hours.

PCR- PCR is critical in several operations. In particular at the very end of the DES calculation,one try to extract the small amount of DNA that corresponds to the one correct description out of 2^{56}. Sequencing the DNA to discover the key requires macro scopic amounts of DNA , so the extracted DNA has to be amplified greatly.

Extracting- The mathematics of DNA assume that if we choose values for any no. of bases, then we could extract all the DNA that had a string of bases at its left end. Unfortunately this not true. Presently the longest patterns one can use are about 500 bases.

Cutting and Reconnecting DNA- At the present , DNA molecules can be cut only at certain patterns of bases.As long as that is true, one has to choose encodings so that the DNA can be cut where and only where the algorithms requires.

Very large scale production of restriction enzymes, ligases and affinity columns- DNA computers solving interesting problems using current algorithms will use incredibly large amount of expensive material such as ligases ,restriction aenzymes, affinity columns etc. Methods to greatly lower the production casts of these materials , and to produce them in enormous quantities relative to current biotechnology applications, are required in order to envision a feasible DNA Computer.

V. APPLICATION OF DNA COMPUTATION

1. Massively Parallel Processing

The primary advantage offered by most proposed models of DNA based computation is the ability to handle millions of operations in parallel. The massively parallel processing capabilities of DNA computers may give them the potential to find tractable solutions to otherwise intractable problems, as well as potentially speeding up large, but otherwise solvable, polynomial time problems requiring relatively few operations. The use of DNA to perform massive searching and related algorithms will be referred to as "classic" DNA computation for the purposes of this discussion.

2. Solving NP-Complete and Hard Computational Problems

After Adleman's and Lipton's initial results, much of the work on DNA computing has continued to focus on solving NP-complete and other hard computational problems. Problems in NP-complete are such that there is no polynomial time solution known to exist using conventional computer algorithms. That is, as the complexity of these problems increase, the time required to solve them increases at an exponential rate. These problems are also said to be *intractable*, but, if within the domain of NP-complete a *tractable* solution can be found to one of these problems then it can also be used to solve for all other problems in the set.

3. Storage and Associative Memory

DNA might also be used to mirror, and even improve upon, the associative capabilities of the human brain. In [4] Baum proposed a method for making a large content addressable memory using DNA. A truly content addressable memory occurs when a data entry can be directly retrieved from storage by entering an input that most closely resembles it over other entries in memory. This input may be very incomplete, with a number of wildcards, and in an associative memory might even contain bits that do not actually occur within the closest match. This contrasts with a conventional computer memory, where the specific address of a word must be known to retrieve it. Rather, the use of this technique would replicate what is thought by many to be a key factor in human intelligence.

4. DNA2DNA Applications

Another area of DNA computation exists where conventional computers clearly have no current capacity to compete. This is the concept of DNA2DNA computations as suggested in [12] and identified as a potential *killer app*. DNA2DNA computations involve the use of DNA computers to perform operations on unknown pieces of DNA without having to sequence them first. This is achieved by re-coding and amplifying unknown strands into a redundant form so that they can be operated on according to techniques similar to those used in the sticker model of DNA computation. Many of the errors inherent in other models of DNA computing can hopefully be ignored in DNA2DNA computing because there will be such a high number of original strands available for operations.

5. Implications to Biology, Chemistry, and Medicine

While the development of DNA computational methods may have many directly applicable applications, the biggest contribution of research in this area may be much more fundamental and will likely fuel many indirect benefits. In many papers, including [2] and [16] it is stressed that high levels of collaboration between academic disciplines will be essential to affect progress in DNA computing. Such collaboration may very well lead to the development of a DNA computer with practical advantages over a conventional computer but has an even greater likelihood of contributing to an increased understanding of DNA and other biological mechanisms. The need for additional precision could effect progress in biomolecular techniques by placing demands on bio-chemists and their tools that might not otherwise be considered.

6. DNA's Role in Computer Science

The study of DNA as both a natural storage medium, and as a tool of computation, could shed new light on many areas of computer science. Despite the high error rates encountered in DNA computing, in nature DNA has little understood but resilient mechanisms for maintaining data integrity. By studying genetic code for these properties, new principles of error correction may be discovered, having use within conventional electronic computation in addition to the new biomolecular paradigms

VI. DNA COMPUTING IMPLEMENTATIONS

The majority of models for DNA computing in the current literature are primarily theoretical. Most have been designed by computer scientists and mathematicians with the goal of proving that universal computation can be achieved by DNA computing. Although this is clearly a critical property for any successful model of DNA computing, it is not necessarily the best starting point for practical DNA computing applications. In this section we examine models inspired by biological techniques that lend themselves much more readily to implementation using existing methods and technologies.

Whiplash PCR, proposed in Kiga et al. (1997) and Sakamoto et al. (2000), is based on the classical model of a state machine. The state machine consists of a single-stranded DNA molecule with the subsequence at the 3^0 end of the molecule being considered the current state of the machine. Since the molecule is single-stranded, it will naturally attempt to bond with another complementary strand of DNA to form a stable double-stranded molecule. In the absence of other molecules, and if a self-complementary subsequence exists, the single strand will form a bond with itself, creating what is known as a hairpin structure Figure .

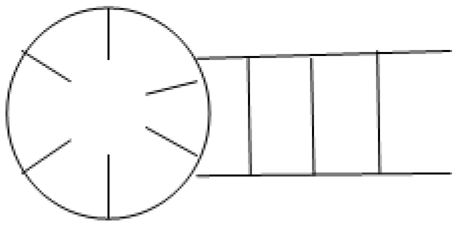
This allows one to encode the state machine’s transition table in the molecule itself in the form stop–newstate–oldstate. This creates a situation where if the 3^0 head of the DNA molecule (the current state) is complementary to the sequence representing oldstate then the two will anneal, forming a new molecule. At this point, the enzyme DNA polymerase will attempt to extend the old state toward the new state.

Although it is possible to perform single-molecule computations with such a system, its real power lies in the ability to have many such molecular state machines operating in parallel. Eric Winfree (1998) presents a modification of this system that is capable of solving some NP-complete problems in O(1) biosteps (although, of course, the space requirements make solving large instances impractical) using parallelization.

In contrast to whiplash PCR, which is solution-based, the model of surface-based DNA computing emerged, which is dependent upon the DNA strands being affixed to a physical surface. Liu et al. (2000) present a method for solving the satisfiability problem (SAT) that involves working with single-stranded DNA affixed to a gold surface.

The first step involves generating the entire solution set for the given problem (with respect to the chosen encoding) and then attaching each potential solution DNA single strand to the glass surface. We work here with the example given in Liu et al. (2000). Each base in a given single strand represents a single bit, A and C for 0, G and T for 1.

Marking is accomplished by synthesizing the set of DNA strands that are complementary to the strings in which bit i has value b and allowing these strands to anneal to the surface-affixed strands. The destroy-marked and destroy-unmarked operations are easily accomplished by washing the surface with a solution containing enzymes known as exonucleases that will specifically destroy either single-stranded or double-stranded DNA. Unmark can be accomplished by applying distilled water to the surface. The low salinity of the water will destabilize the double-stranded DNA and cause it to separate into single strands again. Finally, the test for emptiness is implemented by removing the DNA from the surface, amplifying it with PCR, and checking for a product.



VII. SUMMARY AND CONCLUSIONS

In this article we have presented a brief survey of several research areas in the field of DNA computing. We presented a number of formal models as well as implementations of biomolecular computing, where biomolecules and biochemical reactions are used to perform specific computations. The advantage of this mode of computation over more traditional methods (e.g., electronic computing) is the promise of massive parallelism that would be unachievable in silicon. Clearly, biomolecular computing is not an appropriate solution to every computational problem, given the amount of overhead involved in setting up a reaction protocol. However, as laboratory techniques mature, DNA computing may become an attractive alternative to electronic computers for difficult problems that would benefit from a parallel approach.

In addition to the in vitro approach, more interest is now being placed on in vivo computing. The ability to ‘‘program’’ a living cell offers not only interesting computational possibilities, but also perhaps a new tool for medical and biological sciences. If one could accurately utilize the genetic mechanisms of a cell for accomplishing man-made tasks, it is theoretically possible that a collective of such cells could be used as biological nanomachines to carry out many of the desirable medical functions described in the current nanotechnology literature (Freitas, 2001).

In an era when the term biotechnology is used loosely by many, DNA computing offers the possibility of true biotechnology that will allow large, difficult computations to be performed in reasonable time; possibly the ability to utilize cellular function at the genetic level; new tools and methods for analyzing the overwhelming amount of genetic information that global efforts in gene sequencing and analysis are producing; and most significantly, a formal mathematical framework for understanding the underlying mechanisms of cellular genetics.

REFERENCES

[1] Adleman, L. 1994. Molecular computation of solutions to combinatorial problems. Science 266:1021–1024.
[2] Balan, M. S., K. Krithivasan, and Sivasubramanyam. 2001. DNA Computing (DNA-Based Computers 7), LCNS 2340 (Lecture Notes in Computer Science), N. Jonoska, N. C. Seeman, Eds., pp. 290–299. Berlin: Springer-Verlag.
[3] Baranda, A. V., F. Arroyo, J. Castellanos, and R. Gonzalo. 2001. DNA Computing (DNA-Based Computers 7), LCNS 2340 (Lecture Notes in Computer Science), N. Jonoska, N. C. Seeman Eds., pp. 350–359. Berlin: Springer, Verlag.
[4] Benenson, Y., T. Paz-Elizur, R. Adar, Z. Livneh, E. Keinan, and E. Shapiro. 2001. Programmable and autonomous computing machine made out of biomolecules. Nature 414:430–434.
[5] Bennett, C. H., and R. Landauer. 1985. The fundamental physical limits of computation. Sci. Am. 253(1):48–56 (intl. ed. 38–46).
[6] Bloom, B., and C. Bancroft. 1999. Liposome-mediated biomolecular computation. In DNA Based Computers V, eds. E. Winfree and D. Gifford, pp. 39–48. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 54. Providence Rhode Island: Amer-ican Mathematical Society Press.
[7] Braich, R. S., N. Chelyapov, C. Johnson, P. W. K. Rothemund, and L. Adleman. 2002. Solution of a 20-variable 3-SAT problem on a DNA computer. Science, 296:499–502.
[8] Calladin, C. R., and H. R. Drew. 1999. Understanding DNA: The Molecule and How It Works. San Diego: Academic Press.

[9] Csuhaj-Varju, E., R. Freund, L. Kari, and G. Paun. 1996. DNA computing based on splicing: Universality results. In First Annual Pacific Symposium on Biocomputing, eds. L. Hunter and T. E. Klein, pp. 179–190. Singapore: World Scientific.
[10] Csuhaj-Varju, E., L. Kari, and G. Paun. 1996. Test tube distributed systems based on splicing. Comput. AI 15:211–232.
[11] Culik, K., and T. Harju. 1991. Splicing semigroups of dominoes and DNA. Discrete Appl. Math. 31:261–277.
[12] Daley, M., L. Kari, R. Siromoney, and G. Gloor. 1999. Circular contextual insertion=deletion with applications to biomolecular computation. Proc. 6th Int. Symp. String Processing and Information Retrieval, Cancun Mexico, 47–54.
[13] Dassow, J., and V. Mitrana. 1996. Splicing grammar systems. Comput. AI 15:109–122.
[14] Deninnghoff, K. L., and R. W. Gatterdam. 1989. On the undecidability of splicing systems. Int. J. Comput. Math. 27:133–145.
[15] Engelfriet, J., and G. Rozenberg. 1980. Fixed point languages, equality languages, and repre-sentation of recursively enumerable languages. J. ACM 27:499–518.
[16] Freitas, R. A., Jr. 2001. Nanomedicine, vol. I. Georgetown, VA: Landes Bioscience.
[17] Freund, R. 1995. Splicing systems on graphs. Proc. Intelligence in Neural and Biological Systems, 189–194.
[18] Freund, R., L. Kari, and G. Paun. 1999. DNA computing based on splicing: The existence of universal computers. Theory Comput. Syst. 32:69–112.
[19] Gatterdam, R. 1989. Splicing systems and regularity. Int. J. Comput. Math. 31:63–67.
[20] Gatterdam, R. 1994. DNA and twist free splicing systems. In Words, Languages and Combinatorics II, eds. M. Ito and H. Ju¨rgensen, pp. 170–178. Singapore: World Scientific.
[21] Hagiya, M., H. Uejima, and S. Kobayashi. 2001. Horn clause computation by self assembly of DNA molecules. Proc. Seventh Int. Workshop on DNA Based Computers, Tampa, 63–74.
[22] Hartmanis, J. 1995. On the weight of computations. Bull. Eur. Assoc. Theoret. Comput. Sci. 55:136–138.
[23] Head, T. 1987. Formal language theory and DNA: An analysis of the generative capacity of recombinant behaviors. Bull. Math. Biol. 49:737–759.
[24] Head, T. 1992. Splicing schemes and DNA. In: Lindenmayer Systems—Impacts on Theoretical Computer Science, Computer Graphics and Developmental Biology, pp. 371–383. Berlin: Springer-Verlag.
[25] Head, T., G. Paun, and D. Pixton. 1996. Language theory and genetics. Generative mechanisms suggested by DNA recombination. In Handbook of Formal Languages, vol. 2, eds. G. Rozenberg and A. Salomaa, pp. 295–360. Berlin: Springer-Verlag.
[26] Hopcroft, J., J. Ullman, and R. Motwani. 2001. Introduction to Automata Theory, Languages, and Computation, 2nd ed. Reading, MA: Addison–Wesley.
[27] Kari, L. 1997. DNA computing: Arrival of biological mathematics. Math. Intelligencer 19(2):9–22. Kari, L. 1991. On Insertions and Deletions in Formal Languages. Ph.D. thesis, University of Turku, Finland.
[28] Kari, L., S. Konstantinidis, E. Losseva, and G. Wozniak. 2001. Sticky-Free and Overhang-Free DNA Languages, submitted.
[29] Kari, L., and G. Thierrin. 1996. Contextual insertions=deletions and computability. Inform. Comput. 131:47–61.
[30] Kiga, D., K. Sakamoto, M. Hagiya, M. Arita, and S. Yokoyama. 1997. Towards parallel eva-luation and learning of Boolean m-formulas with molecules. In DNA Based Computers III, eds. H. Rubin and D. H. Wood. pp. 57–72. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 48. Providence: American Mathematical Society Press.
[31] Lagoudakis, M., and T. LaBean. 1999. 2D DNA self-assembly for satisfyability. In DNA Based Computers V, eds. E. Winfree and D. Gifford, 141–154. DIMACS Series in Discrete Mathe-matics and Theoretical Computer Science, vol. 54. American Mathematical Society Press.
[32] Landweber, L., and L. Kari. 1999. Universal molecular computation in ciliates. In Evolution as Computation. Eds. L. F. Landweber, E. Winfree. Berlin: Springer-Verlag.
[33] Laun, E., and K. J. Reddy. 1997. Wet splicing systems. In DNA Based Computers III, eds. H. Rubin and D. H. Wood, pp. 73–84. DIMACS Series in Discrete Mathematics and The-oretical Computer Science, vol. 48. Providence: American Mathematical Society Press.
[34] Lipton, R., D. Faulhammer, A. Cukras, and L. Landweber. 1999. When the knight falls: On constructing an RNA computer. In DNA Based Computers V, eds. E. Winfree and D. Gifford, 1–8. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 54. American Mathematical Society Press.
[35] Liu, Q., L. Wang, A. G. Frutos, A. E. Condon, R. M. Corn, and L. M. Smith. 2000. DNA computing on surfaces. Nature 403:175–179.