DNA Computation: *Applications and Perspectives*

Somnath Tagore [1]*, Saurav Bhattacharya[2], Md Ataul Islam[3] and Md Lutful Islam[4]

[1]*Department of Bioinformatics, Dr. D.Y. Patil University, Navi Mumbai, India*
[2]*Department of Molecular Biology & Biotechnology, University of Kalyani, India*
[3]*Department of Chemical Technology, University of Calcutta, India*
[4]*Department of Computer Science, MH Saboo Siddique College of Engg., Mumbai, India*

Abstract

The computational capability of living systems has intrigued researchers for years. Primarily, the focus has been on implementing aspects of living systems in computational devices. Computer literal peoples expand their hand to the molecular biologist and chemist to explore the potential for computation of biological molecules line Deoxyribonucleic Acid (DNA) and Ribonucleic Acid (RNA) which are information carrying molecules. In this context, DNA computation is basically a collection of specially selected DNA strands whose combinations will result in the solution to some problems. DNA computation rather DNA-based computing is at the intersection of several threads of research. Main advantages of DNA computation are miniaturization and parallelism over conventional silicon-based machines. The information-bearing capability of DNA molecules is a cornerstone of modern theories of genetics and molecular biology. In this paper we have tried to focus on some key issues regarding the used and implementation DNA-based devices in life science field. We have also tried to suggest its advantage over silicon computers.

Introduction

Biochemical "nanocomputers" already exist in nature. They are manifested in all living things. But they're largely uncontrolled by humans. We cannot, for example, program a tree to calculate the digits of pi (Melkikh, 2008; Ogasawara et al., 2008; Watson et al., 1987). The idea of using DNA to store and process information took off in 1994 when a California scientist firstly used DNA in a test tube to solve a simple mathematical problem (Stryer, 1995). Since then, several research groups have proposed designs for DNA computers, and those attempts have relied on an energetic molecule called ATP for fuel. "This re-designed device uses its DNA input as its source of fuel," said Ehud Shapiro, who led an Israeli research team (Von Neumann, 1966). To the naked eye, the DNA computer looks like clear water solution in a test tube. There is no mechanical device. A trillion bio-molecular devices could fit into a single drop of water. Instead of showing up on a computer screen, results are analyzed using a technique that allows scientists to see the length of the DNA output molecule. DNA computation is a form of computing which uses DNA and molecular biology, instead of the traditional silicon-based computer technologies. A single gram of DNA with volume of 1 cm³ can hold as much information as a trillion compact discs, approximately 750 terabytes (Ulam, 1972; Holland, 1992; Feynman, 1961).

DNA-based computing is at the intersection of several threads of research. The information bearing capability of DNA molecules is a cornerstone of modern theories of genetics and molecular biology. The information in a DNA molecule is contained in the sequence of nucleotide bases, which hydrogen bond in a complementary fashion to form double- stranded molecules from single-stranded oligonucleotides (Wu, 2001). Various aspects of life inspired early results in computer science in the 1950's (J. von Neumann's universal constructor and computer (Head, 1987), S. Ulam's models of growth using cellular automata. A second development occurred in the early 1970's with J. Holland's computational implementation of fundamental biological mechanisms, such as genetic operations (splicing, recombination and mutation) and evolution (Dailey et al., 2009). Finally, a third stage inaugurated by L. Adleman's 1994 proof of concept that recombinant properties of real DNA can actually use massive parallelism to solve problems appropriately encoded into single DNA strands.

History of DNA Computation

The computational capability of living systems has intrigued researchers for years. Primarily, the focus has been on implementing aspects of living systems in computational devices. Examples are cellular automata, genetic algorithms, artificial neural networks, and artificial life. The argument has been that universal computational devices are capable of simulating the behavior of physical, living systems through appropriate programming. Therefore, the direction of innovation has been from biology to computer science. Rechard Feynman first introduced the molecular computation (Heyries et al., 2009) at early 1960s. This field was initially developed by Leonard Adleman of the University of Southern California. In 1994, Adleman demonstrated a proof-of-concept use of DNA as form of computation which was used to solve the seven-point Hamiltonian path problem. The electronic computers use two digits that are 0 and 1 known as binary digits, whereas a DNA strand contains four-letter alphabet that is A, T, G and C which can hold much more information than earlier type of computers. Since the initial Adleman experiments, advances have been made, and various Turing machines have been proven to be constructable. Lipton proposed DNA experiments to solve the satisfiability problem. In 1997, Ouyan et al. presented a molecular biology based experimental solution to the "maximal clique" problem. In 2000, Liu et al. designed a DNA model system, where a multibased encoding stategy is used in an approach to surface-based DNA computation. In 2001, Wu analyzed and improved their

surface-based method (Macko and Whelan, 2008). All their works use the tools of molecular biology, and all demonstrate the feasibility of carrying out computations at the molucular level. One of the formal frameworks for molecular computations is the Head's splicing system, which gives a theoretical foundation for computing based on DNA recombination (Liu et al., 2008). In the year 2004, Shapiro and co-workers constructed a DNA computer, coupled with an input and output module and is capable of diagnosing cancerous activity within a cell, and then releasing an anti-cancer drug upon diagnosis (Łoś et al., 2008).

DNA vs Silicon Micro-Processor

Main advantages of DNA computation are miniaturization and parallelism over conventional silicon-based machines. For example, a square centimeter of silicon can currently support around a million transistors, whereas current manipulation techniques can handle to the order of 1020 strands of DNA. However, this miniaturization alone does not give us the computational power that DNA promises (Feyen et al., 2008). Beyond these DNA computers have some drawbacks like the gel electrophoresis and polymerase chain reaction [PCR] are slower by a factor of 108 compared with operations on conventional computers. This drawback is outweighed by the potential for the massive parallelism offered by DNA computation, due to the fact that all strands are manipulated simultaneously. The combination of parallelism and miniaturization promises orders of magnitude more operations per second than current supercomputers (Chang et al., 2008).

DNA, with its unique data structure and ability to perform many parallel operations, allows you to look at a computational problem from a different point of view. Transistor-based computers typically handle operations in a sequential manner. Of course there are multi-processor computers, and modern CPUs incorporate some parallel processing, but in general, in the basic von Neumann architecture computer, instructions are handled sequentially. A von Neumann machine, which is what all modern CPUs are, basically repeats the same "fetch and execute cycle" over and over again. It fetches an instruction and the appropriate data from main memory and it executes the instruction (Na et al., 2008). It does these many, many times in a row, really, really fast. The great Richard Feynman, in his Lectures on Computation, summed up von Neumann computers by saying, "the inside of a computer is as dumb as hell, but it goes like mad!" DNA computers, however, are non-von Neuman, stochastic machines that approach computation in a different way from ordinary computers for the purpose of solving a different class of problems (Leclerc et al., 2008).

Typically, increasing performance of silicon computing means faster clock cycles (and larger data paths), where the emphasis is on the speed of the CPU and not on the size of the memory. For example, will doubling the clock speed or doubling your RAM give you better performance? For DNA computation, though, the power comes from the memory capacity and parallel processing. If forced to behave sequentially, DNA loses its appeal. For example, let's look at the read and write rate of DNA. In bacteria, DNA can be replicated at a rate of about 500 base pairs a second. Biologically this is quite fast (10 times faster than human cells) and considering the low error rates, an impressive achievement (Kim et al., 2008). But this is only 1000 bits/sec, which is a snail's pace when compared to the data throughput of an average hard drive. But look what happens if you allow many copies of the replication enzymes to work on DNA in parallel.

First of all, the replication enzymes can start on the second replicated strand of DNA even before they're finished copying the first one. So already the data rate jumps to 2000 bits/sec. But look what happens after each replication is finished - the number of DNA strands increases exponentially (2n after n iterations). With each additional strand, the data rate increases by 1000 bits/sec. So after 10 iterations, the DNA is being replicated at a rate of about 1Mbit/sec. After 30 iterations it increases to 1000 Gbits/sec. This is beyond the sustained data rates of the fastest hard drives (Bakar et al., 2008).

Now let's consider how you would solve a nontrivial example of the traveling salesman problem (number of cities > 10) with silicon vs. DNA. With a von Neumann computer, one naive method would be to set up a search tree, measure each complete branch sequentially, and keep the shortest one. Improvements could be made with better search algorithms, such as pruning the search tree when one of the branches you are measuring is already longer than the best candidate. A method you certainly would not use would be to first generate all possible paths and then search the entire list (Han et al., 2008). Why? Well, consider that the entire list of routes for a 20 city problem could theoretically take 45 million GB of memory (18! routes with 7 byte words)! Also for a 100 MIPS computer, it would take two years just to generate all paths (assuming one instruction cycle to generate each city in every path). However, using DNA computation, this method becomes feasible! Also, routes no longer have to be searched through sequentially. Operations can be done all in parallel (Tominaga et al., 2007).

## Computational Complexity and Problems

Mathematical biology is a highly interdisciplinary area that lies at the intersection of mathematics and biology. Certain stochastic processes and statistical methods have been developed to solve problems in various branches of biology. Despite the complexity of the technology involved the idea behind Mathematical biology is the simple observation that the following two processes one biological and one mathematical are analogous: -

1. The very complex structure of a living being is the result of applying simple operations, copying, splicing etc to initial information encoded in a DNA sequence.
2. The result f (w) of applying a computable function to an argument w can be obtained by applying a combination of basic simple functions to w (Lin et al., 2007; Henkel et al., 2007; Chen et al., 2007; Cost and Cozzarelli, 2007).

A single strand of DNA can be likened to a string consisting of a combination of four different symbols A, G, C and T. Mathematically, this means we have at our disposal a 4 letter alphabet $\Sigma = \{A, G, C, T\}$ to encode information, which is more than enough considering that an electronic computer needs only two digits 1 and 0 for the same purpose. Some of the simple operations that can be performed on DNA sequences are accomplished by a number of commercially available restriction enzymes that execute a few basic tasks. A remarkable fact about Adleman' s result is that not only does it give a solution to a mathematical problem but that the problem solved is a hard computational problem in the sense explained below (Cost, 2007).

Mathematical models

Problems can be ranked in difficulty according to how long the best algorithm to solve the problem will take to execute on a single computer. Algorithms can be divided into a number of classes on the basis of time complexity (Figure 1).

Figure 1: Time complexity.

Figure 2: Space complexity.
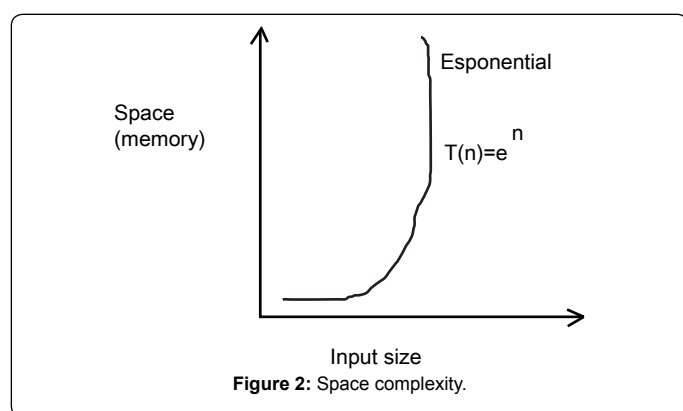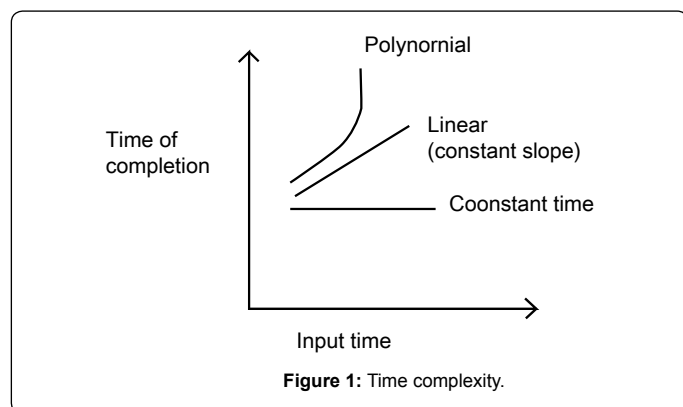
1. Linear: - These have a time complexity such as $T(n) = a*n + k$, 'a' is a constant.
2. Constant Time Algorithm: - These have a time complexity such as $T(n) = k$.
3. Polynomial Algorithm (Quadratic relationship): - These have a time complexity such as $T(n) = n^2 + a * n$ 'a' is a constant
4. Exponential: - These are very complex having time complexity such as $T(n) = e^n$.

Algorithms whose running time is bounded by a polynomial (respectively exponential) function, in terms of the size of the input describing the problem, are in the "polynomial time" class P (respectively the exponential time class EXP). A special class of problems, apparently intractable including P and included in EXP is the "non-deterministic polynomial time" class, or NP [24] (Figure 2). The following inclusions between classes of problems hold:

$$P \in NP \in EXP \in Universal$$

NP contains the problems for which no polynomial time algorithm solving them is known, but that can be solved in polynomial time by using a non-deterministic computer. The directed Hamiltonian path problem is a special kind of problem in NP known as NP-complete. An NPcomplete problem has the property that every other problem in NP can be reduced to it in polynomial time (Li et al., 2006; Condon, 2006; Ibrahim, 2006).

Aldeman's hypothesis: In 1994, Leonard M. Adleman solved an unremarkable computational problem with a remarkable technique. The type of problem that Adleman solved is formally known as a directed Hamiltonian Path (HP) problem, but is more popularly recognized as a variant of the so-called "traveling salesman problem." A Hamiltonian Path in a connected graph is defined as a closed walk

that traverses every vertex of graph 'G' exactly once, except the starting vertex at which the walk also terminates (Feldkamp et al., 2006; Li et al., 2005; Grover and Mathies, 2005).

The above Figure 3 shows a connected graph having a Hamiltonian path as ABCDEFGH. The path passes through each vertex exactly once. The Traveling Salesman Problem can be stated as: A salesman is required to visit a number of cities during a trip. Given the distances between the cities, in what order should he travel so as to visit ever city precisely once and return home, with the minimum mileage traveled? Aldeman's work is significant for a number of reasons (Shortreed et al., 2005; Cardona et al., 2005; Kim et al., 2005; Yang and Yang, 2005).

1. It illustrates the possibilities of using DNA to solve a class of problems that is difficult or impossible to solve using traditional computing methods.
2. It is an example of computation at a molecular level, potentially a size limit that may never be reached by the semiconductor industry.
3. It demonstrates unique aspects of DNA as a data structure.
4. It demonstrates that computing with DNA can work in a massively parallel fashion (Tanaka et al., 2005; Lee et al., 2004; Liu et al., 2004).

DNA has the unique ability to carry out multitasking operations and perform large number of functions simultaneously. Transistor-based computers typically follow the basic von Neumann architecture where instructions are handled sequentially. A von Neumann machine repeats the same "fetch and execute cycle" over and over again; it fetches an instruction and the appropriate data from main memory, and it executes the instruction. DNA computers are non-von Neumann in nature and are stochastic machines that approach computation in a different way from ordinary computers for the purpose of solving a different class of problems. For DNA computation, though, the power comes from the memory capacity and parallel processing (Schmidt et al., 2004; Halpin and Harbury, 2004).

Shortcomings of aldeman's experiment: The complexity of the traveling salesman problem simply doesn't disappear when applying a different method of solution, it still increases exponentially. Regarding the power of computation while using this method, Adleman mentioned some of these features. A typical desk top computer can execute approximately 106 operations per second. The fastest super computers currently available can execute approximately 1012 operations per second. If the concatenation of two DNA molecules is considered as a single operation and if it is assumed that about half of the approximately 4×1014 edge oligonucleotides in Step 1

Figure 3: Hamiltonian path.
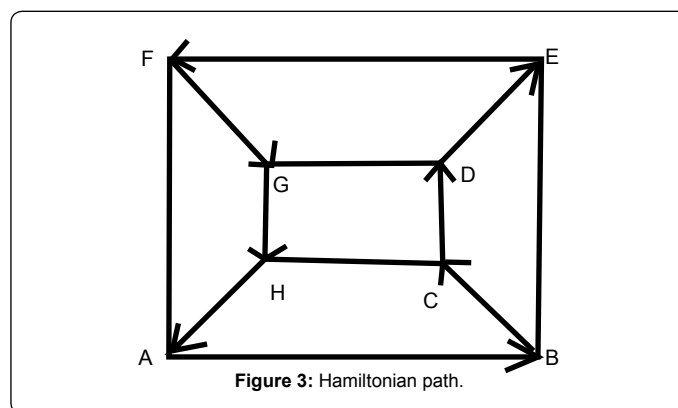
were ligated, then during step1 approximately 1014 operations were executed (Zeng et al., 2007). At this scale, the number of operations per second during the ligation step would exceed that of current super computers by more than a thousand fold. Thus the potential of molecular computation is impressive. What is not clear is whether such massive numbers of inexpensive operations can be productively used to solve real computational problems. One major advantage of electronic computers is the variety of operations they provide and the flexibility with which these operations can be applied. However, for certain intrinsically complex problems such as the directed Hamiltonian path problem where existing electronic computers are very inefficient and where massively parallel searches can be organized to take advantage of the operations that molecular biology currently provides, it is conceivable that molecular computation might compete with electronic computation in the near future (Wang et al., 2000).

For Aldeman's method, what scales exponentially is not the computing time, but rather the amount of DNA. Unfortunately this places some hard restrictions on the number of cities that can be solved. After the Adleman article was published, more than a few people have pointed out that using his method to solve a 200 city HP problem would take an amount of DNA that weighed more than the earth. Another factor that places limits on his method is the error rate for each operation. These operations are not deterministic but stochastically driven each step contains statistical errors, limiting the number of iterations you can do successively before the probability of producing an error becomes greater than producing the correct result (Su and Smith, 2004; Feldkamp et al., 2004). For certain specialized problems, DNA computers are faster and smaller than any other computer built so far. But DNA computation does not provide any new capabilities from the standpoint of computability theory, the study of which problems are computationally solvable using different models of computation. For example, if the space required for the solution of a problem grows exponentially with the size of the problem (EXPSPACE problems) on von Neumann machines it still grows exponentially with the size of the problem on DNA machines. For very large EXPSPACE problems, the amount of DNA required is too large to be practical.

DNA logic gates

Logic gates are a vital part of how your computer carries out functions that you command it to do. These gates convert binary code moving through the computer into a series of signals that the computer uses to perform operations. Currently, logic gates interpret input signals from silicon transistors, and convert those signals into an output signal that allows the computer to perform complex functions. The Rochester team's DNA logic gates are the first step toward creating a computer that has a structure similar to that of an electronic personal computer. Instead of using electrical signals to perform logical operations, these DNA logic gates rely on DNA code. These gates are actually tiny DNA processing centers that detect specific fragments of the genetic blueprint as input, and then splice together the fragments to form a single output. For instance, a genetic gate called the "And gate" links two DNA inputs by chemically binding them so they're locked in an end-to-end structure, similar to the way two Legos might be fastened by a third Lego between them. The researchers believe that these logic gates might be combined with DNA microchips to create a breakthrough in DNA computation (Ogihara and Ray, 2000).

SAT problems

The satisfiability (SAT) problem is a core problem in mathematical logic and computing theory. This has been used in solving many problems that involves automated reasoning, computer-aided design, computeraided manufacturing, machine vision, database, robotics, integrated circuit design, computer architecture design, and computer network design. In recent years, many optimization methods, parallel algorithms, and practical techniques have been developed for solving SAT (Gu et al., 1997). Lipton extended the work of Adleman in solving any NP complete problems directly using biological experiments. Since the biological machines will be limited in the amount of parallelism that they can perform, solving a SAT problem on a large number of variables directly is far better than using the reduction from SAT to Hamiltonian Path problem (Lipton, 1995). Lipton also suggested methods to speed up computations.

Computational Power of DNA

DNA computers

Though double stranded DNA appears to be a good, stable storage medium for information, most proposed DNA computation systems use single stranded DNA (oligonucleotides) for storage (and computation). This choice is largely because these proposals depend on the annealing or hybridization of oligonucleotides to perform data storage or computation actions. Unfortunately hybridization is imprecise, and incorrect hybridizations easily occur. This places strict requirements on codeword formation and puts an upper bound on the amount of information which can be stored (Li et al., 2004).

The church-turing thesis: It states that no realizable computing device can be more powerful than a Turing machine. One of the main reasons that Church-Turing's thesis is widely accepted is that very diverse alternate formalizations of the class of effective procedures have all turned out to be equivalent to the Turing machine formalization. These alternate formalizations include Markov normal algorithms, Post normal systems, type 0 grammars etc (Li et al., 2003).

Mapping of a subset of the Cartesian power set N^n into N, where $n >= 1$ and N is the set of natural numbers, are referred as partial functions. If the domain of such a function equals N^n, then the function is called total. E. g.

1. The zero function: $Z(x_0) = 0$, for all $x_0 \in N$.
2. The successor function: $S(x_0) = x_0 + 1$, for all $x_0 \in N$.
3. The projection function: For all I, n and $x_i \in N$, $0 <= i <= n$ is $U_i^{n+1}(x_0, x_1, \ldots\ldots\ldots,x_n) = x_i$ (Liu et al., 2003).

A function f is defined partial-recursively if it is the zero function, the successor function, or a projection function. It is defined by composing functions which are defined partial recursively. It is defined by the recursion scheme from functions which are defined partial recursively. It is defined using the minimization operation on a function that is defined partial recursively and is total. It was proved that a function f is partial recursive if and only if there is a Turing machine which computes the values of f. On the other hand, according to the Church Turing thesis, everything that can be effectively computed by any kind of device can be computed by a Turing machine. As a consequence, partial recursive functions came to be known also under the name of effectively computable functions (Liu et al., 2003; Zhang et al., 2003; ZhiXiang et al., 2003).

Implementation techniques: Despite the progress achieved in DNA computation, the main obstacles to creating a practical DNA computer still remain to be overcome. These obstacles are roughly of two types:

1. Practical: Arising primarily from difficulties in dealing with large scale systems and in coping with errors.

2. Theoretical: Concerning the versatility of DNA computers and their capacity to efficiently accommodate a wide variety of computational problems (Hug and Schuler, 2003).

The synthesis of a DNA strand can sometimes result in the strand annealing to itself and creating a hairpin structure. Even the seemingly straightforward mixing operation can sometimes pose problems. If DNA is not handled gently, the sheer forces from pouring and mixing will fragment it. Also of concern for this operation is the amount of DNA which remains stuck to the walls of the tubes, pumps, pipette tips etc and is thus lost from the computation. Hybridization has also to be carefully monitored because the thermodynamic parameters necessary for annealing to occur are sequence dependent. Hybridization stringency refers to the number of complementary base pairs that have to match for DNA oligonucleotides to bond. Separation of strands by length and extraction of strands containing a given pattern can also be inefficient and this might pose problems with scale-up of the test tube approach (Penchovsky and Ackermann, 2003; Kishi et al., 2003).

Besides the accuracy of bio-operations, another peril of the implementation of DNA computations is the fact that the size of the problem influences the concentration of reactants and this in turn has an effect on the rate of production and quality of final reaction products. An analysis of Adleman's experiment showed that an exponential loss of concentration occurs even on sparse digraphs and that this loss of concentration can become a significant consideration for problems not much larger than those solved by Adleman. The idea of algorithm transformation can be suggested in cases where the DNA algorithm amounts to sorting a huge library of initial candidate solution complexes into those which encode a solution to a problem and those which do not. In such cases, including Adleman's experiment the main occurring errors are false positives, false negatives and strand losses. A possible solution to these types of problems is to change the algorithms by using intelligent space and time trade offs (Liu et al., 2002; Yin et al., 2002; Wang et al., 2001).

The transformation makes use of a slowdown factor of M, proposing for a given algorithm 'A' a new algorithm 'A' ' that has a smaller error rate than A as follows:

1. Repeat M times.
2. Run A on input I, producing tubes Y and N.
3. Discard tube N and rename tube Y as tube I.
4. Return tube I as the "Yes" tube and an empty tube as "No". This approach is of value when the original algorithm A is known to place very reliably good sequences into the "Yes" tube i.e. low false negatives but too often also place bad sequences into the "Yes" tube i.e. high false positives (Hug and Schuler, 2001).

Error debugging approach: A corresponding variation of the above transformation can be used if the algorithm is known to have high false negatives and low false positives. This and similar methods are used in the model employs stickers that are short complementary sequences to mark the "on" bits, while the unmarked bits are considered "off". The advantage of the proposed sticker model is that it does not rely on short lived materials as enzymes and that it does not involve processes that are costly in terms of energy such as PCR. To put things in perspective, an opposite approach that relies only on PCR to solve problems has proved to be less error prone (Darehmiraki, 2009; Zhang et al., 2006).

Satisfiability problem: A potential DNA experiment can be described for finding a solution to another NP-complete problem that can be called Satisfiability Problem. The Satisfiability Problem consists of a Boolean expression, the question being whether or not there is an assignment of truth values, true or false to its variables that makes the value of the whole expression true. DNA algorithms have since been proposed for expansion of symbolic determinants, graph connectivity and knapsack problem using dynamic programming, road coloring problem, matrix multiplication, addition, exascale computer algebra problems etc (Marathe et al, 2001; Penchovsky et al., 2000).

Data encryption standard

Data Encryption Standard (DES) encrypts 64 bit messages and uses a 56 bit key. Breaking DES means that given one (plain-text, cipher-text) pair, we can find a key which maps the plain-text to the cipher-text. A conventional attack on DES would need to perform an exhaustive search through all of the 256 DES keys, which, at a rate of 100,000 operations per second would take 10, 000 years. Thus the molecular programs came into existence that takes about 4 months of laboratory work instead (Cho, 2000).

It is estimated that DNA computation could yield tremendous advantages from the point of view of speed, energy efficiency and economic information storing. E.g. in Adleman's model, the number of operations per second could be up to approximately 1.2×10^{18}. This is approximately 1,200,000 times faster than the fastest supercomputer (Rothemund, 2000). DNA computers also have the potential for extraordinary energy efficiency. In principle, one joule is sufficient for approximately 2×10^{19} 1igation operations. This is remarkable considering that the second law of thermodynamics dictates a theoretical maximum of 34×10^{19} (irreversible) operations per joule (at room temperature). Existing supercomputers are far less efficient, executing at most 10^9 operations per joule. Finally, storing information in molecules of DNA could allow for an information density of approximately 1 bit per cubic nanometer, while existing storage media store information at a density of approximately 1 bit per 10^{12} nm^3 (Ogihara and Ray, 2000).

Operability of DNA-based devices

From a practical point of view it maybe not that important to simulate a Turing machine by a DNA computation device. One should not aim to fit the DNA model into the Procrustean bed of classical models of computation, but try to completely rethink the notion of computation. DNAbased devices have been addressed for most models of DNA computation that have so far been proposed (Sakakibara and Suyama, 2000; Faulhammer et al., 2000). The existing models of DNA computation are based on various combinations of a few primitive biological operations: -

1. Synthesize: Synthesizing a desired polynomial length strand used in all models.
2. Mixing: Pour the contents of the test tubes into another one to achieve union of bases.
3. Annealing: Bond together two single stranded complementary DNA sequences by cooling the solution.
4. Melting: Break apart a double stranded DNA into its single stranded complementary components by heating the solution (Lee et al., 1999).
5. Amplifying: Copying, make copies of DNA strands by using the Polymerase Chain Reaction (PCR).
6. Separation: Separating the strands by length using gel electrophoresis.
7. Extraction: Extracting those strands that contain a given

pattern as a substring by using affinity purification (Wang et al., 1999).

8. Cutting: Cutting DNA double strands at specific sites by using restriction enzymes.

9. Ligation: Paste DNA strands with compatible sticky ends by using DNA ligase.

10. Substitution: Substitute, insert or delete DNA sequences by using PCR site specific oligonucleotide mutagenesis.

11. Marking: Marking single strands by hybridization, complementary sequences are attached to the strands, making them double stranded.

12. Destroying: Destroying the marked strands by using exonucleases.

13. Detecting: Reading the given the contents of a tube and say 'yes' if it contains at least one DNA strand and 'no' otherwise (Aoi et al., 1999; Mills et al., 1999; Fu and Beigel, 1999).

The bio-operations listed above are used to write programs which receive a tube containing DNA strands as input and return as output either 'yes' or 'no' or a set of tubes.

Non-determinism

DNA computation suggests the potential of DNA with immense parallelism and huge storage capacity by solving NP-complete problems such as the Traveling Salesman Problem (TSP). If DNA computation is used to solve TSP, however, weights between vertexes in sequence design cannot be encoded effectively. Studies propose an algorithm for code optimization (ACO) that applies DNA coding method to DNA computation in order to encode weights in TSP effectively. By applying ACO to TSP, we designed sequence more effectively than Adleman's DNA computation algorithm. Furthermore, we could find the shortest path quickly and reduce biological error rate (Beigel and Fu, 1997).

A decision problem is in the class NP if: (a) there is no known algorithm for it that will execute in polynomial time on a conventional computer and (b) it can be solved in polynomial time using a non-deterministic computer (one with inherently unlimited parallelism). A problem is intractable if no polynomial time algorithm can possibly be devised for it. A problem in NP is NP-complete if every other problem in NP can be expressed in terms of it by means of a polynomial time algorithm. NP-complete problems are considered to be the hardest problems, since if any problem in NP is shown to be intractable then all NP-complete problems are intractable. However, if any NP-complete problem can be solved in polynomial time, all problems in NP become tractable (Beigel and Fu, 1998).

A non-deterministic computer can be simulated by a test tube of DNA strands using the techniques of molecular biology to implement operations on them. In theory, many otherwise intractable problems can be implemented in polynomial time on such a DNA computer. In fact, Adleman's simulation of the Hamiltonian Path problem, known to be NP-complete, has an execution time that is linear in the number of vertices of the graph. This is the reason that DNA computation has become a new focus of research in computer science. Following Adleman's success, polynomial time algorithms for other NP-complete problems, such as SAT, Independent Set and 3-colorability have been published. So far, however, a number of problems have made the implementation of DNA algorithms impractical for all but small instances of the problems.

Technologies for DNA Computation

DNA word sets

For generating sets non-interacting DNA sequences an algorithm

is developed that employs various thermodynamic models for duplex stabilities and secondary structures prediction. DNA 'word' structure is used where individual 'words' of a typical given length are concatenated into longer sequences. Word sets are generated and their figures of merit are compared to sets as described previously in the literature. This is followed by verifying the predicted hybridization behavior using standard UV hyperchromism measurements of duplex melting temperatures (Shortreed et al., 2005).

Markov chains

Various algorithms performing computations over Markov chains have been developed. These determine sequence power of the transition matrix of a Markov chain and their properties of convergence. Some other algorithms help enable to estimate this limit. These also allow the computation of a limit using DNA computation. The states and the transition probabilities have been encoded using strands of DNA for generating paths of the Markov chain (Cardona et al., 2005).

Sequence complexity

It has been noticed that randomly generated oligonucleotide populations serve as pools for selecting non-cross-hybridizing sequences, for nanoscale self-assembly and biological and biomedical applications, as well as for DNA computation applications. Various nonlinear kinetic models are present for the complexity estimation of large unknown polynucleotide populations. Models are used to estimate the sequence complexity of the random 20 base-pair population after in vitro renaturation experiments. The kinetic behaviors of the random 20mers can also be evaluated with in vitro thermal melting experiments (Garzon et al., 1999).

Sticker model

It is essentially easier creating an initial data pool covering answers at first place followed by a series of selection process to destroy the incorrect ones. The surviving DNA sequences are read as the solutions to the problem. But, algorithms are limited to the problem size. As the number of parameters in the studied problem grows, the algorithm becomes impossible owing to the tremendous initial data pool size. The solution sequences are built in parts to satisfy one clause in a step, and eventually solve the whole Boolean formula after a number of steps. The size of data pool grows from one sort of molecule to the number of solution assignments (Yang and Yang, 2005).

Traveling salesman problem

DNA encoding method can be used to represent numerical values and a biased molecular algorithm based on the thermodynamic properties of DNA. DNA strands are designed to encode real values by variation of their melting temperatures. The thermodynamic properties of DNA are used for effective local search of optimal solutions using biochemical techniques, such as denaturation temperature gradient polymerase chain reaction and temperature gradient gel electrophoresis. The algorithm is applied to the traveling salesman problem, an instance of optimization problems on weighted graphs (Kim et al., 2005).

Molecular nanotechnology

Researchers have discovered DNA sequences and structures with new functional properties for preventing the expression of harmful genes. Bioinformaticians design rigid DNA structures that serve as scaffolds for the organization of matter at the molecular scale, and

can build simple DNA-computing devices, diagnostic machines and DNA motors. The integration of biological & engineering advances offers great potential for therapeutic & diagnostic applications (Condon, 2006).

Cellular computing

Cells perform computation by reading and writing DNA using processes that modify sequences at the DNA or RNA level. Two ciliated protozoans of the genus Oxytricha had solved a potentially harder problem using DNA several million years earlier. The solution to this problem, which occurs during the process of gene unscrambling, represents one of nature's ingenious solutions to the problem of the creation of genes. RNA editing can also be viewed as a computational process offering a second algorithm for the construction of functional genes from encrypted pieces of the genome (Landweber and Kari, 1999).

Autonomous DNA computation

A one-pot autonomous DNA computation machine is proposed that is based on photochemical gate transition. Here, photoligation via 5-carboxyvinyldeoxyuridene (cvU) containing oligodeoxynucleotides and photocleavage via carbazole - modified oligodeoxynucleotides, were employed. The binary digit additions are carried out by one-time irradiation at 366 nm in the single test tube. The fluorescence readout by the DNA chip was in good agreement with the correct answer of binary digit additions (Ogasawara et al., 2008).

Clustering approaches

Clustering is used for revealing a structure in highly dimensional data and arriving at a collection of meaningful relationships in data and information granules. DNA computation has also been used for developing clustering techniques. This is very useful while dealing with huge data sets, unknown number of clusters and encountering a heterogeneous character of available data. It is shown the essential components of the clustering technique through the corresponding mechanisms of DNA computation (Bakar et al., 2008).

Application

Splicing system model of dna recombination

The *Splicing System model* introduced by Tom Head aims to be a one pot computer with all the operations carried out in principle by enzymes. Moreover, it has the theoretical advantage of being a mathematical model with all the claims backed up by mathematical proofs (Lee et al., 2004; Manca et al., 1999).

1. An alphabet is a finite nonempty set its elements are called letters or symbols. $\sum^*$ denotes the free monoid generated by the alphabet $\sum$ under the operation of catenation and juxtaposition.
2. The elements of $\sum^*$ are called words or strings. The empty string (the null element of $\sum^*$) is denoted by λ. A language over the alphabet $\sum$ is a subset of $\sum^*$. E.g. if $\sum = \{a, b\}$ then aaba, aabbb $= a^2b^3$ are words over $\sum$ and the following sets are languages over$\sum$: $L_1 = \{\lambda\}$, $L_2 = \{a, ba, aba, abbaa\}$, $L_3 = \{a^p | p \text{ prime}\}$.
3. The catenation of languages L_1 and L_2 is denoted by $L_1 L_2$ is defined by $L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}$ (Landweber and Kari, 1999; Maley, 1998).
4. A finite language can always be defined by listing all of its words. Such a procedure is not possible for infinite languages and therefore other devices for the representation of infinite languages have been developed. One of them is to introduce

a generative device and define the language as consisting of all the words generated by the device. The basic generative devices used for specifying languages are grammars (Liu et al., 1998).

5. A generative grammar is an ordered quadruple $G = \{N, T, S, P\}$, where N and T are disjoint alphabets, $S \in N$ and P is a finite set of ordered pairs (u, v) such that u, v are words over $N \cup T$ and u contains at least one letter of N. The elements of N are called non-terminals and those of T terminals, S are called the axiom. Elements $\{u, v\}$ of P are called rewriting rules and are written u $=>$v. If $x = x_1ux_2$, $y = x_1vx_2$, and $u => v \in P$, then we write $x => y$ and say that x derives y in the grammar G. The reflexive and transitive closure of the derivation relation $=>$ is denoted by $=>^*$. The language generated by G is $L\{G\} = \{w \in T^* \mid S =>^*w\}$.
6. Grammars are classified by imposing restrictions on the forms of productions (Kulić, 1998). A grammar is called of type-0 if no restriction (zero restrictions) is applied to the rewriting rules and is called regular if each rule of P is of the form $A\rightarrow aB$, $A\rightarrow a$, $A,B \in N$, $a \in T$. The family of finite languages will be denoted by FIN_ the family of languages generated by regular grammars by REG and the family of languages generated by type-0 grammars by L_0(Roweis et al., 1998).

Given an alphabet $\sum$ and two strings x and y over$\sum$, the splicing of x and y according to the splicing rule r consists of two steps:

1. Cut x and y at certain positions determined by the splicing rule r
2. Paste the resulting prefix of x with the suffix of y respectively the prefix of y with the suffix of x. Using the formalism splicing rule r over $\sum$ is a word of the form $\alpha_1 \# \beta_1 \$ \alpha_2 \# \beta_2$ where $\alpha_1, \beta_1, \alpha_2, \beta_2$ are strings over $\sum$ and $\#$, $\$$ are markers not belonging to $\sum$.
3. We say that z and w are obtained by splicing x and y according to the splicing rule $r = \alpha_1 \# \beta_1 \$ \alpha_2 \# \beta_2$ and we write $(x, y) \rightarrow_r (z, w)$ if and only if
$x = x_1 \alpha_1 \beta_1 x'$ $z = x_1 \alpha_1 \beta_2 y'_2$
$y = y_2 \alpha_2\beta_2y_2'$ $w = y_2 \alpha_2\beta_1x_1'$ for some $x_1, x_1', y_2,y_2' \in \sum*$ (Freund et al., 1998; Frutos et al., 1997).
4. The words $\alpha_1\beta_1$ and $\alpha_2\beta_2$ are called sites of the splicing, while x and y are called the terms of the splicing. The splicing rule r determines both the sites and the positions of the cutting: between α_1 and β_1 for the first term and α_2 and β_2 for the second.
5. The site $\alpha_1\beta_1$ can occur more than once in x while the site $\alpha_2\beta_2$ can occur more than once in y.Whenever this happens, the sites are chosen non-deterministically. As a consequence, the result of splicing x and y can be a set containing more than one pair (z. w) (Ouyang et al., 1997).
6. How the splicing works can be illustrated by using it to simulate the addition of two positive numbers, n and m. If an alphabet $\sum= \{a, b, c\}$ is considered and the splicing rule $r = a \# b \$ c \# a$, then the splicing of $x = a^n b$ and $y = c a^m$ according to r yields the words a^{n+m} and cb.
7. The splicing operation can be used as a basic tool for building a generative mechanism, called splicing system.
8. If the classical notion of a set is used, we implicitly assume that after splicing x and y and obtaining z and w, we may use again x or y as terms of the splicing i.e. the strings are not consumed by splicing. Similarly, there is no restriction on the number of copies of the newly obtained z and w (Gibbons et al., 1997; Csuhaj-Varjú et al., 1996).

DNA nanomachines

DNA has been explored as an excellent material for building large-scale nanostructures, constructing individual nanomechanical

devices, and performing computations (Pool, 1995). A variety of DNA nanomechanical devices have been previously constructed that demonstrate motions such as open/close, extension/contraction, andmotors/rotation (Morimoto et al., 2007), mediated by external environmental changes such as the addition and removal of DNA fuel strands or the change of ionic composition of the solution (Chee et al., 1996). The DNA walker could ultimately be used to carry out computations and to precisely transport nanoparticles of material. The walker can be programmed in several ways in this direction. For example, information can be encoded in the walker fragments as well as in the track so that, while performing motion, the walker simultaneously carries out computation. Yin et al in the year 2005 designed an autonomous DNA walking device in which a walker moves along a linear track unidirectionally (Liu et al., 2000). Sherman and Seeman in 2004 have constructed a DNA walking device controlled by DNA fuel strands. Reif in the year 2003 designed an autonomous DNA walking device and an autonomous DNA rolling device that move in a random bidirectional fashion along DNA tracks. Shin and Pierce in 2004 designed the DNA walker for molecular transport. Recently, Yin et al in 2005 encoded computational power into a DNA walking device embedded in a DNA lattice and therefore accomplished the design for an autonomous nano-mechanical device capable of universal computation and translational motion (Kaplan et al., 2001).

Other models

The construction of unusual DNA molecules has a long history (Seeman et al., 1996). Such molecules include knots (Du and Seeman, 1994), Holliday junctions (Fu et al., 1994) and octahedra (Zhang and Seeman, 1994). The construction of such objects relies upon the creation of branched DNA molecules known as junctions. Because these objects do not occur naturally, they must be engineered in the laboratory. In order to determine the specific sequences to assign to components of branches (i.e. single strands of DNA) a technique known as sequence symmetry minimization is used (Seeman, 1982). This assigns sequences to various components such that when they are placed in solution they hybridize to form the desired structure. Winfree and coworkers in 1996 proposed the tendency of DNA structures to self-assemble as a computational tool (Winfree et al., 1996). They show that the self-assembly of complex branches known as double crossovers. Although of great theoretical interest, experimental investigations of the power of self-assembly are still at a very preliminary stage. They also showed that, in principle, the two dimensional self-assembly model is experimentally implementable.

DNA-based reversible gates

Due to the progress made in areas of nanotechnology, storing information in terms of DNA computing-chips has been possible. The use of reversible logic gates for synthesis of circuits in aspects of quantum computing is being tried (Tumpane et al., 2007). Furthermore, it has been seen that in this article, the reversible logic is proposed to be simulated by using DNA molecules and bio-chemistry operations: the input and the output of a reversible gate or a reversible sequential circuit are both DNA sequences, and the computing progresses correspond to the biochemistry operations. These can also be used for designing the optimal reversible sequential circuits, some new trends in DNA and quantum computing (Benenson et al., 2003).

Conclusion and Perspectives

The apparent ease with which DNA hybridization can be formalized made Adleman's invention very attractive to researchers in the fields of computer science and discrete mathematics. Numerous architectures for DNA based computing have since been proposed. There is no doubt that DNA will soon exhibit all sorts of eccentric behavior that cannot be conveniently incorporated into formal descriptions. Some researchers are therefore skeptical whether DNA will ever follow bacteriorhodpsin on its (narrow) path to take on the silicon competition. However, DNA may very well have potential for soft computing applications that can tolerate or even benefit from components which do not overly conform to formal rules.

You can use your DNA computer only to solve a specific problem and you can typically use it only once, in one problem case. DNA computers may solve problems with an alarmingly high error rate. Instead of being "universal", DNA computers are "instance" computers: they are good at one instance of a problem, and one problem type. If you set up a DNA computer to simulate a universal computer that can be programmed easily, you still lose out against the above trade-off, so it is probably a waste of time to try to build Turing machines with any hope of utilising parallelism to speed them up.

The advantages of DNA computers, at least for the time being, are outweighed by the difficulties of using them to do anything useful. There is more theory than practical success at the moment, though they have enormous potential for applications in medicine, farming, food and fingerprint recognition and forensic science. Conventional problems are best handled by conventional computers; with DNA computers, we need to think differently. One trick is to use molecules to solve problems they are naturally good at solving. Thus DNA computers are good at doing DNA-related things.

Another example is to make a fish-freshness sensor. You must measure many compounds and assess odour and taste to tell if a fish is fresh. This sort of problem is one that animals solve all the time, so there is good reason to hope that DNA computers could be built to do it, too.

The field of DNA computation remains alive and promising, even as new challenges emerge. Most important among these are the uncertainty, because of the DNA chemistry, in the computational results, and the exponential increase in number of DNA molecules necessary to solve problems of interesting size. Despite these issues, definite progress has been made both in quantifying errors, and in development of new protocols for more efficient and error- tolerant DNA computation. In addition, new paradigms based on molecular evolution viz. molecular systematics, mutationism etc. have emerged from molecular biology to inspire new directions such as molecular computers, which are capable of doing high performance computing in DNA computation.

References

1. Aoi Y, Yoshinobu T, Tanizawa K, Kinoshita K, Iwasaki H (1999) Ligation errors in DNA computation. Biosystems 52: 181-187.

2. Bakar RB, Watada J, Pedrycz W (2008) DNA approach to solve clustering problem based on a mutual order. Biosystems 91: 1-12.

3. Beigel R, Fu B (1997) Molecular computing, bounded nondeterminism, and efficient recursion. Proceedings of the 24th International Colloquium on Automata, Languages, and Programming.

4. Beigel R, Fu B (1998) Solving intractable problems with DNA computation. Annual IEEE Conference on Computational Complexity.

5. Benenson Y, Adar R, Paz-Elizur T, Livneh Z and Shapiro E (2003) DNA molecule provides a computing machine with both data and fuel. Proc Natl Acad Sci USA 100: 2191-2196.

6. Cardona M, Colomer MA, Conde J, Miret JM, Miró J, et al. (2005) Markov chains: computing limit existence and approximations with DNA. Biosystems 81: 261-266.

7. Chang YJ, Hu CY, Yin LT, Chang CH, Su HJ (2008) Dividable membrane with multi-reaction wells for microarray biochips. J Biosci Bioeng 106: 59-64.

8. Chee M, Yang R, Hubbell E, Berno A, Huang XC, et al. (1996) Accessing genetic information with high- density DNA arrays. Science 274: 610-614.

9. Chen P, Li J, Zhao J, He L, Zhang Z (2007) Differential dependence on DNA ligase of type II restriction enzymes: a practical way toward ligase-free DNA automaton. Biochem Biophys Res Commun 353: 733-737.

10. Cho A (2000) DNA computation. Hairpins trigger an automatic solution. Science 288: 1152-1153.

11. Condon A (2006) Designed DNA molecules: principles and applications of molecular nanotechnology. Nat Rev Genet 7: 565-575.

12. Cost GJ (2007) Enzymatic ligation assisted by nucleases: simultaneous ligation and digestion promote the ordered assembly of DNA. Nat Protoc 2: 2198-2202.

13. Cost GJ, Cozzarelli NR (2007) Directed assembly of DNA molecules via simultaneous ligation and digestion. Biotechniques 42: 86-89.

14. Csuhaj-Varjú E, Freund R, Kari L, Păun G (1996) DNA computation based on splicing: universality results.

15. Dailey MM, Hait C, Holt PA, Maguire JM, Meier JB, et al. (2009) Structure-based drug design: from nucleic acid to membrane protein targets. Exp Mol Pathol 86: 141-150.

16. Darehmiraki M (2009) A new solution for maximal clique problem based sticker model. Biosystems 95: 145-149.

17. Du SM, Seeman NC (1994) The construction of a trefoil knot from a DNA branched junction motif. Biopolymers 34: 31-37.

18. Faulhammer D, Lipton RJ, Landweber LF (2000) Fidelity of enzymatic ligation for DNA computation. J Comput Biol 7: 839-848.

19. Feldkamp U, Wacker R, Schroeder H, Banzhaf W, Niemeyer CM (2004) Microarray-based in vitro evaluation of DNA oligomer libraries designed in silico. Chemphyschem 5: 367-372.

20. Feldkamp U, Schroeder H, Niemeyer CM (2006) Design and evaluation of single-stranded DNA carrier molecules for DNA-directed assembly. J Biomol Struct Dyn 23: 657-666.

21. Feyen O, Lueking A, Kowald A, Stephan C, Meyer HE, et al. (2008). Off-target activity of TNF-alpha inhibitors characterized by protein biochips. Anal Bioanal Chem 391: 1713-1720.

22. Feynman RP (1961) There's plenty of room at the bottom. In Miniaturization, Gilbert, D. Ed.; Reinhold: New York, pp. 282-296.

23. Freund R, Păun G, Rozenberg G, Salomaa A (1998) Bidirectional sticker systems. Pac Symp Biocomput. 535-546.

24. Frutos AG, Liu Q, Thiel AJ, Sanner AM, Condon AE, et al. (1997) Demonstration of a word design strategy for DNA computation on surfaces. Nucleic Acids Res 25: 4748-4757.

25. Fu B, Beigel R (1999) Length bounded molecular computing. Biosystems 52: 155-163.

26. Fu TJ, Tse-Dinh YC, Seeman NC (1994) Holliday junction crossover topology. J Mol Biol 236: 91-l05.

27. Garzon MH, Jonoska N, Karl SA (1999) The bounded complexity of DNA computation. Biosystems 52: 63-72.

28. Gibbons A, Amos M, Hodgson D (1997) DNA computation. Curr Opin Biotechnol 8: 103-106.

29. Grover WH, Mathies RA (2005) An integrated microfluidic processor for single nucleotide polymorphism-based DNA computation. Lab Chip 5: 1033-1040.

30. Gu J, Purdom PW, Franco J, Wah BW (1997) Algorithms for the Satisfiability (SAT) Problem: A Survey. In DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society.

31. Halpin DR, Harbury PB (2004) DNA display I. Sequence-encoded routing of DNA populations. PLoS Biol 2: E173.

32. Han A, Zhu D, Pan J (2008) DNA solution based on sequence alignment to the Minimum Spanning Tree problem. Int J Bioinform Res Appl 4: 188-200.

33. Head T (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bull Math Biol 49: 737-759.

34. Henkel CV, Bäck T, Kok JN, Rozenberg G, Spaink HP (2007) DNA computation of solutions to knapsack problems. Biosystems 88: 156-162.

35. Heyries KA, Blum LJ, Marquette CA (2009) Straightforward protein immobilization using redox-initiated poly(methyl methacrylate) polymerization. Langmuir 25: 661- 614.

36. Holland JH (1992) Adaptation in Natural and Artificial Systems. MIT Press: Cambridge.

37. Hug H, Schuler R (2001) Strategies for the development of a peptide computer. Bioinformatics 17: 364-368.

38. Hug H, Schuler R (2003) Measurement of the number of molecules of a single mRNA species in a complex mRNA preparation. J Theor Biol 221: 615-624.

39. Ibrahim Z, Tsuboi Y, Ono O (2006). Hybridization-ligation versus parallel overlap assembly: an experimental comparison of initial pool generation for direct-proportional length-based DNA computation. IEEE Trans Nanobioscience 5: 103-109.

40. Kaplan PD, Thaler DS, Libchaber A (2001) Parallel overlap assembly of paths through a directed graph. In Preliminary Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, pp. 127-141.

41. Kim JS, Lee JW, Noh YK, Park JY, Lee DY, et al. (2008) An evolutionary Monte Carlo algorithm for predicting DNA hybridization. Biosystems 91: 69-75.

42. Kim JW, Carpenter DP, Deaton R (2005) Estimating the sequence complexity of a random oligonucleotide population by using in vitro thermal melting and Cot analyses. Nanomedicine 1: 220-230.

43. Kishi N, Ogino M, Saito I, Yoshimura Y, Fujimoto, K (2003) Photochemical ligation of DNA "words" for DNA computation. Nucleic Acids Res Suppl 3: 183-184.

44. Kulić IM (1998) Evaluating polynomials on the molecular level--a novel approach to molecular computers. Biosystems 45: 45-57.

45. Landweber LF, Kari L (1999) The evolution of cellular computing: nature's solution to a computational problem. Biosystems 52: 3-13.

46. Leclerc E, Kirat K, Griscom L (2008) In situ micropatterning technique by cell crushing for co-cultures inside microfluidic biochips. Biomed Microdevices 10: 169-177.

47. Lee CM, Kim SW, Kim SM, Sohn U (1999) DNA computation the Hamiltonian path problem. Mol Cells 9: 464-469.

48. Lee JY, Shin SY, Park TH, Zhang BT (2004) Solving traveling salesman problems with DNA molecules encoding numerical values. Biosystems 78: 39-47.

49. Li D, Huang H, Li X, Li X (2003) Hairpin formation in DNA computation presents limits for large NP-complete problems. Biosystems 72: 203-207.

50. Li D, Li X, Huang H, Li X (2005) The surface-based approach for DNA computation is unreliable for SAT. Biosystems 82: 20-25.

51. Li D, Li X, Huang H, Li X (2006) Scalability of the surface-based DNA algorithm for 3-SAT. Biosystems 85: 95-98.

52. Li Y, Tseng YD, Kwon SY, D'Espaux L, Bunch JS, et al. (2004) Controlled assembly of dendrimer-like DNA. Nat Mater 3: 38-42.

53. Lin CH, Cheng HP, Yang CB, Yang CN (2007) Solving satisfiability problems using a novel microarray-based DNA computer. Biosystems 90: 242-252.

54. Lipton RJ (1995) Using DNA to solve NP-complete problems. Priceton University.

55. Liu L, He Y, Zhang Y, Ma S, Ma H, et al. (2008) Parallel scan spectral surface plasmon resonance imaging. Appl Opt 47: 5616-5621.

56. Liu Q, Frutos AG, Thiel AJ, Corn RM, Smith LM (1998) DNA computation on surfaces: encoding information at the single base level. J Comput Biol 5: 269-278.

57. Liu Q, Guo Z, Condon AE, Korn RM, Lagally MG, et al. (2000) A surface based approach to DNA computation. In Proceedings of the Second Annual Meeting on DNA Based Computers, 206-216.

58. Liu Q, Wang L, Frutos AG, Condon AE, Corn RM, et al. (2000) DNA computation on surfaces. Nature 403: 175-179.

59. Liu W, Gao L, Liu X, Wang S, Xu J (2003) Solving the 3-SAT problem based on DNA computation. J Chem Inf Comput Sci 43: 1872-1875.

60. Liu W, Shi X, Zhang S, Liu X, Xu J (2004). A new DNA computation model for the NAND gate based on induced hairpin formation. Biosystems 77: 87-92.

61. Liu Y, Xu J, Pan L, Wang S (2002) DNA solution of a graph coloring problem. J Chem Inf Comput Sci 42: 524-528.

62. Łoś M, Łoś JM, Wegrzyn G (2008) Rapid identification of shiga toxin-producing Escherichia coli (STEC) using electric biochips. Diagn Mol Pathol 17: 179-184.

63. Macko P, Whelan MP (2008) Fabrication of holographic diffractive optical elements for enhancing light collection from fluorescence-based biochips. Opt Lett 33: 2614- 2616.

64. Maley CC (1998) DNA computation: theory, practice, and prospects. Evol Comput 6: 201-229.

65. Manca V, Martín-Vide C, Păun G (1999) New computing paradigms suggested by DNA computation: computing by carving. Biosystems 52: 47-54.

66. Marathe A, Condon AE, Corn RM (2001) On combinatorial DNA word design. J Comput Biol 8: 201-219.

67. Melkikh AV (2008) DNA computation, computation complexity and problem of biological evolution rate. Acta Biotheor 56: 285-295.

68. Mills AP, Yurke B, Platzman PM (1999) Article for analog vector algebra computation. Biosystems 52: 175-180.

69. Morimoto N, Arita M, Suyama A (2007) Solid phase DNA solution to the Hamiltonian path problem. In Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, PA., pp. 83-92.

70. Na K, Jung J, Kim O, Lee J, Lee TG, et al. (2008) "Smart" biopolymer for a reversible stimuli-responsive platform in cell-based biochips. Langmuir 24: 4917-4923.

71. Ogasawara S, Ami T, Fujimoto K (2008) Autonomous DNA computation machine based on photochemical gate transition. J Am Chem Soc 130: 10050-10051.

72. Ogihara M, Ray A (2000) DNA computation on a chip. Nature 403: 143-144.

73. Ouyang Q, Kaplan PD, Liu S, Libchaber A (1997) DNA solution of the maximal clique problem. Science 278: 446-449.

74. Penchovsky R, Birch-Hirschfeld E, McCaskill JS (2000) End-specific covalent photo-dependent immobilisation of synthetic DNA to paramagnetic beads. Nucleic Acids Res 28: E98.

75. Penchovsky R, Ackermann J (2003) DNA library design for molecular computation. J Comput Biol 10: 215-229.

76. Pool R (1995) A boom in plans for DNA computation. Science 268: 498- 499.

77. Rothemund PW (2000) Using lateral capillary forces to compute by self-assembly. Proc Natl Acad Sci USA 97: 984-989.

78. Roweis S, Winfree E, Burgoyne R, Chelyapov NV, Goodman MF, et al. (1998) A sticker-based model for DNA computation. J Comput Biol 5: 615-629.

79. Sakakibara Y, Suyama A (2000) Intelligent DNA chips: logical operation of gene expression profiles on DNA computers. Genome Inform Ser Workshop Genome Inform 11: 33-42.

80. Schmidt KA, Henkel CV, Rozenberg G, Spaink HP (2004) DNA computation using single-molecule hybridization detection. Nucleic Acids Res 32: 4962-4968.

81. Seeman NC (1982) Nucleic acid junctions and lattices. J Theor Biol 99: 237-247.

82. Seeman NC, Wang H, Liu B (1996) The perils of polynucleotides: the experimental gap between the design and assembly of unusual DNA structures. In Proceedings of the Second Annual Meeting on DNA Based Computers. D/MAC% Series in Discrete Mathematics and Theoretical Computer Science, Providence, RI: American Mathematical Society.

83. Shortreed MR, Chang SB, Hong D, Phillips M, Campion B, et al. (2005) A thermodynamic approach to designing structure-free combinatorial DNA word sets. Nucleic Acids Res 33: 4965-4977.

84. Stryer L (1995) Biochemistry, 3d Ed, New York: W. H. Freeman and Company.

85. Su X, Smith LM (2004) Demonstration of a universal surface DNA computer. Nucleic Acids Res 32: 3115-3123.

86. Tanaka F, Kameda A, Yamamoto M, Ohuchi A (2005) Design of nucleic acid sequences for DNA computation based on a thermodynamic approach. Nucleic Acids Res 33: 903-911.

87. Tominaga K, Watanabe T, Kobayashi K, Nakamura M, Kishi K, et al. (2007) Modeling molecular computing systems by an artificial chemistry - its expressive power and application. Artif Life 13: 223-247.

88. Tumpane J, Kumar R, Lundberg EP, Sandin P, Gale N, et al. (2007) Triplex addressability as a basis for functional DNA nanostructures. Nano Lett 7: 3832-3839.

89. Ulam S (1972) On some mathematical problems connected with patterns of growth of gures. Essays on cellular automata, 219-231.

90. Von Neumann J (1966) Theory of Self-Reproducing Automata, University of Illinois Press: Urbana, IL.

91. Wang L, Liu Q, Frutos AG, Gillmor SD, Thiel AJ, et al. (1999) Surface-based DNA computation operations: DESTROY and READOUT. Biosystems 52: 189-191.

92. Wang L, Rodriguez-Tomé P, Redaschi N, McNeil P, Robinson A, et al. (2000). Accessing and distributing EMBL data using CORBA (common object request broker architecture). Genome Biol 1: RESEARCH0010.

93. Wang L, Hall JG, Lu M, Liu Q, Smith LM (2001) A DNA computation readout operation based on structure-specific cleavage. Nat Biotechnol 19: 1053-1059.

94. Watson JD, Hopkins NH, Roberts JW, Steitz JA, Weiner AM (1987) Molecular Biology of the Gene. (5th edn), The Benjamin Cummings Publishing Co.

95. Winfree E, Yang X, Seeman NC (1996) Universal computation via selfassembly of DNA: some theory and experiments. In Proceedings of the Second Annual Meeting on DNA Based Computers. D/MAC% Series in Discrete Mafbemafics and Theoretical Computer Science, Providence, RI: American Mathematical Society.

96. Wu H (2001) An improved surface-based method for DNA computation. Biosystems 59: 1-5.

97. Yang CN, Yang CB (2005) A DNA solution of SAT problem by a modified sticker model. Biosystems 81: 1-9.

98. Yin Z, Zhang F, Xu J (2002) A Chinese Postman Problem based on DNA computation. J Chem Inf Comput Sci 42: 222-224.

99. Zeng E, Mathee K, Narasimhan G (2007) IEM: an algorithm for iterative enhancement of motifs using comparative genomics data. Comput Syst Bioinform Conf 6: 227-235.

100. Zhang M, Cheng MX, Tarn TJ (2006) A mathematical formulation of DNA computation. IEEE Trans Nanobioscience 5: 32-40.

101. Zhang Y, Seeman NC (1994) The construction of a DNA truncated octahedron. J Am Chem Soc 116: 1661-l669.

102. Zhang ZZ, Zhao J, He L (2003) Progress in molecular biology study of DNA computer. Yi Chuan Xue Bao 30: 886-892.

103. ZhiXiang Y, Fengyue Z, Jin X (2003) The general form of 0-1 programming problem based on DNA computation. Biosystems 70: 73-78.