01 Jan 2006

A Mathematical Formulation of DNA Computation

Mingjun Zhang

Maggie Xiaoyan Cheng
Missouri University of Science and Technology, chengm@mst.edu

Tzyh-Jong Tarn

A Mathematical Formulation of DNA Computation

Mingjun Zhang, *Member, IEEE*, Maggie X. Cheng, *Member, IEEE*, and Tzyh-Jong Tarn, *Fellow, IEEE*

***Abstract*— DNA computation is to use DNA molecules for information storing and processing. The task is accomplished by encoding and interpreting DNA molecules in suspended solutions before and after the complementary binding reactions. DNA computation is attractive, due to its fast parallel information processing, remarkable energy efficiency, and high storing capacity. Challenges currently faced by DNA computation are (1) lack of theoretical computational models for applications, and (2) high error rate for implementation. This paper attempts to address these problems from mathematical modeling and genetic coding aspects. The first part of this paper presents a mathematical formulation of DNA computation. The model may serve as a theoretical framework for DNA computation. In the second part, a genetic code based DNA computation approach is presented to reduce error rate for implementation, which has been a major concern for DNA computation. The method provides a promising alternative to reduce error rate for DNA computation.**

***Index Terms*— DNA computation, mathematical formulation, error rate, genetic code.**

I. Introduction

In the late 1950's, the Nobel laureate Richard Feynman first introduced the idea of computation at a molecular level. In 1994, the concept of DNA computation was demonstrated using experiments to solve a directed Hamiltonian Path Problem (HPP) by Adleman [2]. Since then, the possibility of DNA computation has attracted many researchers' attention.

DNA computation is to use DNA molecules for information storing and processing by encoding and interpreting DNA molecules in suspended solutions before and after DNA complementary binding reactions. The central idea of DNA computation is the Watson-Crick model of DNA structure, which specifies complementary binding properties of DNA molecules.

It is well-known that within cells of any living species, there is a substance called Deoxyribonucleic Acid (DNA), which is a double-stranded helix of nucleotides carrying the genetic information of a cell. This information is the code used within cells to form proteins and is the building block upon which life is formed. A single-stranded DNA consists of a chain of molecules called bases, which protrude from a sugar-phosphate backbone. The four bases are Adenine (A), Thymine (T), Guanine (G), and Cytosine (C). Any single-stranded DNA will adhere tightly to its complementary strand, in which G always pairs with C and A always pairs with T, and vice versa. DNA computation involves to use single-stranded DNA segments to code the problem, let the single-stranded DNA segments react in test tubes or substrate surfaces, and then to find DNA binding strands and interpret the results by applying bio-molecular techniques. This process can be refined into two steps. The first step is to generate all possible solutions to the problem by mixing DNA solutions. DNA complementary binding reactions occur in parallel and extremely fast upon mixing. The second step is to isolate correct solutions through repeated separations of the DNA strands from incorrect solutions and potentially good solutions. A schematic representation of DNA computation is given in Fig. 1, where inputs of the system are single-stranded DNA segments $X, Y, ..., Z$. After hybridization, double-stranded DNA segments $(X...Z)$ that encode possible solutions are detected. In conventional terminologies of computing, the DNA strands may be regarded as hardware. Strand coding can be treated as software. The operating system is to read DNA strands through bio-molecular techniques.

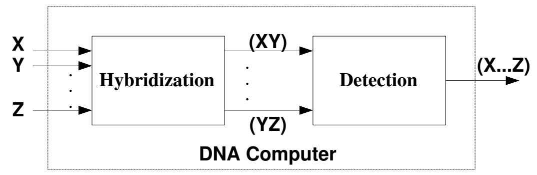


Fig. 1. DNA Computation

DNA computation is attractive mainly for three reasons. First, the computation realizes fast parallel information processing. Second, the process is remarkably energy efficient. Finally, DNA molecules have very high storing capacity. A liter of solution may provide associative memory of up to 10^7 or 10^8 tera-bytes. A DNA strand may need 1000 base pairs to encode a computing processor. So a liter of solution may encode states of approximately 10^{18} distinct processors [13]. DNA computers have been shown to be at least equivalent to a classical Turing machine [1], [10], [14].

A mathematical model is helpful to understand the theoretical aspects of DNA computation. The model is also useful to apply mathematical tools to solve DNA computation problems. Paper [3] proposed a simple abstract model of molecular computers. However, numerous topics related to DNA computation remain open.

In this paper, a mathematical formulation of DNA computation is presented. Based on the formulation, character-based DNA computation is converted into a numerical computation problem. Propositions based on the formulation are also presented. To illustrate the formulation and propose new ideas on solving problems currently faced by DNA computation, a genetic code based DNA computation approach is further

Manuscript received November 18, 2004; accepted November 15, 2005.

Mingjun Zhang (contacting author) is with the Life Sciences and Chemical Analysis Division at Agilent Technologies, California, USA. Email:mingjunzhang@ieee.org. Tel: (408)553-4159.

Maggie X. Cheng is with the Department of Computer Science, University of Missouri, Rolla, Missouri, USA. E-mail: chengm@umr.edu.

Tzyh-Jong Tarn is with the Department of Electrical and Systems Engineering, Washington University in St. Louis, Missouri, USA. E-mail: tarn@wuauto.wustl.edu.

proposed. The genetic code based DNA computation is based on the genetic coding theory of molecular biology. The goal is to reduce the error rate, which has been a major concern for the implementation of DNA computation.

This paper is organized as follows. In Section II, the mathematical formulation of DNA computation is presented. The genetic code based DNA computation is discussed in Section III. Applications of the formulation is given in Section IV. Conclusion and discussion are presented in Section V.

II. A Mathematical Formulation of DNA computation

Define the following notations:

- Let $X = x_i x_{i+1} ... x_j$ and $Y = y_i y_{i+1} ... y_j$ represent single-stranded DNA segments, where $i, j \in N$ and $i \leq j$. N is the natural number. $x_i, y_i \in \{A, T, G, C\}$.
- The complementary sequence of X is defined as $\bar{X}$.
- Let T_i represents the i-th test tube, where $i \in N$. $T_i(+X)$ means that the test tube containing DNA segment X. $T_i(-X)$ means that the test tube T_i does not contain DNA segment X.

DNA computation involves many bio-molecular operations including hybridization, separation, cutting and pasting DNA strands at desired locations. These operations can be generalized at DNA strand level as follows, where "$\rightarrow$" represents the right hand side is the result of the reaction from the left hand side operation.

- Ligation: plus "$+$" operation. Ligation concatenates segments of DNA. Biochemically, it is often invoked after an annealing operation. Although it is possible to use some ligase enzymes to concatenate free-floating double-stranded DNA segments, it is more efficient to allow single-stranded DNA to anneal together, connecting up to a series of single-stranded fragments, and then use ligase to seal the covalent bonds between adjacent fragments. For two single-stranded DNA segments X and Y, a ligation operation can be expressed as "$X + Y \rightarrow [XY]$", where $[XY]$ represents a newly created single-stranded DNA segment.
- Cut: minus "$-$" operation. Restriction enzymes can cut a strand of DNA at a specific address. Some restriction enzymes only cleave single-stranded DNA, while others only cleave double-stranded DNA segments. If a single-stranded DNA X is cut at position n from the $3'$ end of a DNA segment, the process can be expressed as "$-X(n) \rightarrow Y + Z$", where Y and Z are newly generated single-stranded DNA segments and Y has a length of $n - 1$.
- Hybridization: multiplication "$\bullet$" operation. It is a process when single-stranded complementary DNA segments spontaneously form a double-stranded DNA. For single-stranded DNA X and $\bar{X}$, the binding process can be described as "$X \bullet \bar{X} \rightarrow (X\bar{X})$", where $(X\bar{X})$ is a newly created double-stranded DNA. DNA strands enclosed by brackets "(" and ")" are double-stranded DNA and cannot be bonded with other strands unless further melting operation is applied.
- Melting: division "$\backslash$" operation. This is an inverse of the annealing operation. Heating can be selectively used to melt apart short double-stranded DNA segments while leaving longer double-stranded segment intact. For example, "$\backslash(X\bar{X}) \rightarrow X + \bar{X}$" means melting the double-stranded DNA $(X\bar{X})$ as two complementary single-stranded DNA segments X and $\bar{X}$.

One insight from the above formulation is that the DNA computation is very fast. All the above operations are single step DNA molecule reactions, which are extremely fast compared with conventional silicon computation.

In addition to strand level operations, the DNA computation may use the following operations at test tube level, which are performed using sets of DNA segments.

- Mergence: union operation "$\cup$". The operation means two test tubes can be combined, usually by pouring one test tube into the other. For example, "$T_1 \cup T_2 \rightarrow T$" means melting two test tubes T_1 and T_2 together to produce a new test tube T.
- Separation or extraction: difference operation "$-$". The expression "$-(T_i, X) \rightarrow T_j(+X) \cup T_k(-X)$" represents a separation operation applied to the test tube T_i on DNA segment X. The operation produces two test tubes, where the tube T_j contains a string X and the tube T_k does not contain the DNA segment X, where $i, j, k \in N$. Either T_j and T_k could be empty set ϕ. This step is done using gel electrophoresis. It requires the DNA strands to be extracted from the gel once the DNA segments of different length have been identified.
- Amplification: product operation "$\times$". Given a test tube containing DNA strands, the operation is to make multiple copies of a subset of the strands presented. Copies are usually made with PCR. For example, "$\times T \rightarrow T_1 \cup T_2$" means two test tubes T_1 and T_2 containing copies of a subset of DNA strands are produced from the test tube T.
- Detection: question operation "?". This operation means that gel electrophoresis is applied to see if anything of the appropriate length is left within a test tube after PCR amplification. For example, "$T?X \rightarrow$ True" means that the test tube T contains at least one string X. Otherwise, "$T?X \rightarrow$ False".
- Destroy: intersection operation "$\cap$" with an empty set. Subsets of strands can be systematically destroyed or "digested" by enzymes that preferentially break apart nucleotides in either single- or double-stranded DNA segments. The process can be expressed as "$T \cap \phi \rightarrow \phi$".

To further investigate DNA computation as a computational problem, the following concepts are developed.

A. Conversion of character-based DNA sequences to numerical sequences

Three methods are proposed to convert character-based DNA sequences into numerical sequences. One method is to use complex numbers. The second method is to use integer numbers. The third method is to convert DNA sequence into vectors.

1) Complex number representation: Define a function $f(x) : \{A,T,G,C\} \to \{1,-1,i,-i\}$ as

$$f(x) = \begin{cases} 1, & x = A; \\ -1, & x = T; \\ i, & x = G; \\ -i, & x = C. \end{cases} \tag{1}$$

where x is one of the four nucleotides.

The complementary base of each DNA base x can then be calculated by the following inverse function

$$\bar{x} = f^{-1}(-f(x)) = \begin{cases} T, & x = A; \\ G, & x = C; \\ C, & x = G; \\ A, & x = T. \end{cases} \tag{2}$$

By definitions (1) and (2), complementary DNA sequences (either numerical or character-based) can be easily obtained. This means only single-stranded DNA segments need to be specified. The complementary strands can be easily generated using the above functions in either character-based or numerical format.

2) Integer number representation: DNA bases may be mapped as integer numbers as well. Define a function $f(x) : \{A,T,G,C\} \to \{0,1,2,3\}$ as

$$f(x) = \begin{cases} 0, & x = A; \\ 1, & x = C; \\ 2, & x = G; \\ 3, & x = T. \end{cases} \tag{3}$$

Similarly, the complementary base of x can be determined by the following inverse function

$$\bar{x} = f^{-1}(\{3\} - f(x)) = \begin{cases} T, & x = A; \\ G, & x = C; \\ C, & x = G; \\ A, & x = T. \end{cases} \tag{4}$$

where $\{3\}$ represents an appropriate finite length sequence consisting of multiple copies of integer 3. The numerical calculation can then be conducted base by base. For example, the numerical sequence of a DNA segment $X = AGGCAT$ is $f(X) = f(AGGCAT) = 022103$. The complementary segment of X can be easily obtained as $\bar{X} = f^{-1}(\{3\} - f(X)) = f^{-1}(311230) = TCCGTA$.

3) Vector representation: In vector space analysis, numerical value based DNA sequences can be expressed as rows of a matrix. Addition of such kinds of matrices can be regarded as DNA hybridization process. Scalar multiplication produces multiple copies of the sequences in a test tube. Consider the four DNA bases $\{A,T,G,C\}^T$ as a vector, any DNA strand $X = x_1x_2...x_n, n \in N$, can then be expressed as a vector by a transfer matrix Π as

$$X = \Pi \begin{bmatrix} A \\ T \\ G \\ C \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & p_{n3} & p_{n4} \end{bmatrix} \begin{bmatrix} A \\ T \\ G \\ C \end{bmatrix} \tag{5}$$

where $\sum_{j=1}^{4} p_{ij} = 1, \forall i \in N$. Specifically,

$$\begin{aligned} p_{i1} &= \begin{cases} 1, & x_i = A. \\ 0, & otherwise. \end{cases} \\ p_{i2} &= \begin{cases} 1, & x_i = T. \\ 0, & otherwise. \end{cases} \\ p_{i3} &= \begin{cases} 1, & x_i = G. \\ 0, & otherwise. \end{cases} \\ p_{i4} &= \begin{cases} 1, & x_i = C. \\ 0, & otherwise. \end{cases} \end{aligned} \tag{6}$$

In above definition, each row of the matrix Π represents one DNA base. The complementary sequence of $\bar{X}$ can then be obtained by simply swapping column one with column two, and column three with column four as follows

$$\bar{X} = \begin{bmatrix} p_{12} & p_{11} & p_{14} & p_{13} \\ p_{22} & p_{21} & p_{24} & p_{23} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n2} & p_{n1} & p_{n4} & p_{n3} \end{bmatrix} \begin{bmatrix} A \\ T \\ G \\ C \end{bmatrix} \tag{7}$$

For example, the transfer matrix of a single-stranded DNA $X = ACGTGGATCT$ is Π_1 shown in (8). The complementary sequence of X is $\bar{X} = TGCACCTAGA$, whose transfer matrix is Π_2 shown in (8)

$$\Pi_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \Pi_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

The above definitions (6) and (7) make it possible to define a DNA strand as an $n \times 4$ matrix.

The DNA nucleotides may also be defined as a vector directly. For example, $A = [1\ 0]^T$, $T = [-1\ 0]^T$, $G = [0\ 1]^T$, and $C = [0\ -1]^T$. Then a DNA sequence can be expressed as a $2 \times n$ matrix, where n is the number of bases for the DNA sequence. For example, a single-stranded DNA sequence $X = GATCCAGT$ can be expressed as

$$\begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 1 & 0 \end{bmatrix} \tag{9}$$

In a biological process, mutations often occur [7]. A stochastic transfer matrix Γ can be defined as follows to reflect this phenomenon based on the above definitions (6) and (7).

$$\Gamma = \begin{bmatrix} \rho_{aa} & \rho_{at} & \rho_{ag} & \rho_{ac} \\ \rho_{ta} & \rho_{tt} & \rho_{tg} & \rho_{tc} \\ \rho_{ga} & \rho_{gt} & \rho_{gg} & \rho_{gc} \\ \rho_{ca} & \rho_{ct} & \rho_{cg} & \rho_{cc} \end{bmatrix} \tag{10}$$

where ρ_{ij} represents the probability of transformation from DNA base i to j, where $i,j \in \{a,t,g,c\}$. Obviously, $\rho_{ia} + \rho_{it} + \rho_{ig} + \rho_{ic} = 1, \forall i \in \{a,t,g,c\}$. ρ_{ii} is the probability for correct transformation.

The inner product (as the inner product in linear algebra) of a DNA sequence X can then be expressed as $X^T X$, which is a diagonal 2×2 matrix. The first and the last elements in the matrix represent the number of bases in X from the set $\{A, T\}$ or $\{G, C\}$, respectively.

Once a DNA sequence is converted into a numerical sequence, many interesting properties can be investigated. Next, some theoretical results are presented.

B. Some theoretical results

By definition (1) and in viewing a DNA sequence as a vector in the format of (9), the following results are obtained.

Proposition 2.1: If the base-by-base plus operation of two equal-lengthed numerical value based DNA sequences results in a zero vector, then the two DNA sequences are complementary to each other.

Proof: Define vectors $X = [x_1 x_2 ... x_n]^T$ and $Y = [y_1 y_2 ... y_n]^T$ as the two DNA sequences. By the assumption $X + Y$ is equal to a zero vector, we can conclude that $x_j = -y_j, j = 1, ..., n$, where $x_j, y_j \in \{1, -1, i, -i\}$. By definition (1), the sequences are complementary to each other.

Proposition 2.2: If the base-by-base plus operation of all numerical value based DNA sequences in different test tubes results in a zero vector, then the hybridization by mixing the test tubes should be complete. A complete hybridization means all single-stranded DNA sequences find their complementaries.

Proposition 2.3: Under the definition (1), if the inner product of two equal-lengthed sequences is not a real number, then the two sequences are not complementary to each other. It can be further claimed that they are not complementary in G and C.

Proposition 2.4: Under the definition (1), $\forall X, Y \in R^n$ (n represents the number of DNA bases in the strands), if X and Y have a complete hybridization and $X^T Y = 0$, then X and Y have equal number of DNA bases from $\{G, C\}$ and $\{A, T\}$. Similarly, if a complete hybridization occurs, but $X^T Y > 0$, it means that there are more bases from $\{G, C\}$ set than from $\{A, T\}$ set. Otherwise, if complete hybridization occurs, but $X^T Y < 0$, it means that there are more bases from $\{A, T\}$ set.

Proposition 2.5: Under the definition (1), $\forall X, Y \in R^n$ (n represents number of DNA bases in the strands), we have $\|X^T Y\| < \|X\| \|Y\|$, where $\|X^T Y\|$ represents the length of the DNA strand after hybridization. $\|X\|$ and $\|Y\|$ represent the length of single-stranded DNA segments. If a complete hybridization occurs, $\|X^T Y\| = n$. If none of the bases is hybridized, then $\|X^T Y\| = 2n$. This relationship is similar to the well-known Cauchy-Schwartz inequality in linear algebra.

To investigate properties under the above formulation (9) in vector space, the following definitions are proposed.

Definition: Equivalent transfer matrices. Since each single strand of a double-stranded DNA uniquely determines the other strand, each single-stranded DNA can be alternatively used to describe the same DNA double strand. Transfer matrices of a single-stranded DNA and its complementary are regarded as equivalent to each other. For example, Π_1 in (8) is an equivalent transfer matrix of Π_2 in (8) expressed as $\Pi_1 \Leftrightarrow \Pi_2$, and vice versa. Two DNA sequences are complementary to each other, if and only if their transfer matrices are equivalent.

Note, two DNA transfer matrices are equivalent, if and only if one matrix is the result of swapping column one with column two, and column three with column four of the other matrix.

Proposition 2.6: If DNA transfer matrices $A \Leftrightarrow B$ and $B \Leftrightarrow C$, then A is the same as C.

Definition: Similar DNA sequences. Two equal-lengthed DNA sequences that have less than certain percent (usually 10% in practice) different bases in order are regarded as similar sequences. The binding results for similar sequences may be hard to be distinguished using current molecular techniques. It is advised not to use similar sequences to code different words in DNA computation.

Proposition 2.7: **Necessary condition for similar sequences**. Under the formulation (9), if two DNA sequences are similar, then the sum of all columns of the transfer matrices has numerical value variations less than a pre-defined percent of the length of a single DNA sequence.

To code a DNA computation problem, it is critical that all coding words are not similar to each other and they are not complementary to each other. The above propositions can be used to check similarities and complementary properties of DNA sequences for DNA computation.

By the above formulation, the problem of DNA computation can be regarded as the process of finding numerical value based sequences, so that their base-by-base operations (hybridization) results in a zero vector. To illustrate how the above formulation may be used for DNA computation and other related problems, an application section is given following the discussion on genetic coding.

III. Genetic Code based DNA Computation

A. Background Information

DNA computation currently is too error-prone to achieve its great potential. Many ideas of DNA computation assume a zero error rate. In reality, errors appear at every stage. In [2], [9], the problem of high error rates was identified as the most challenging problem for the success of DNA computation. High error resistant method is needed for DNA computation. One open question is whether the error rates in DNA manipulations can be adequately controlled [6], [10]. Some algorithms have been proposed to handle a few of the apparently crippling errors. Paper [8] proposed a surface-based DNA computation algorithm to solve the minimal set cover problem. The technique decreases errors caused by potential DNA strand lost by affixing the DNA onto a silicon surface. In [5], a DNA computation model has been developed that uses dynamic programming and large size of memory available to DNA computers. The goal is to reduce error rates by increasing DNA strands. A more thorough study of decreasing error rates can be found in [6], where methods for making volume decreasing algorithms (the number of strands decreases as the algorithm executes) more resistant to certain types of errors are proposed. One effort in the paper is to convert the decreasing volume problem to a constant volume problem (the number of strands remains the same throughout the computation). The basic idea is to add DNA strand redundancy by increasing solution volume. The technique requires to increase steps of operations and cannot be applied to an algorithm that has constant volume to begin with. The other effort proposed in the paper is to reduce the false negative error rate in the bead separation procedure by double encoding DNA bases. The idea is to have each DNA-encoded base appear twice in separate locations in the strand to increase the possibility of being extracted. However, it is still not clear at present stage whether error rates can be reduced sufficiently to allow a general-purpose DNA computation.

Next, a genetic code based approach is proposed, which appears novel and promising in dealing with error rate reduction. Another nice thing about the proposed approach is that it is not limited to certain type of errors, which have been the cases for most of the methods proposed in the open literature [4]. We start with analyzing the sources of errors.

As discussed earlier, two steps are usually involved in solving a DNA computation problem. Errors usually comes from the following three sources.

- DNA strand extraction. The operations are to remove strands from a test tube containing a given DNA pattern. In reality, only about 95% of the strands matching the pattern can be removed. Sometimes, strands that do not match may be accidentally removed. Even with 99% successful extraction rate, the chance of getting a good strand after multiple steps of extraction exponentially decreases. If only one "solution" is in the test tube, it is almost impossible to exactly extract the strand through a couple of biological reactions and operations.
- Random errors. Random substitution, insertion, and deletion errors may occur in DNA strands, which may lead to wrong binding results.
- PCR errors. The polymerase enzymes do make mistakes when they are synthesizing copies of DNA strands.

The above errors can be reduced by either designing high error resistant coding approach or developing better molecular techniques for later DNA strand extraction. To avoid difficulty in reducing errors at later stages, a method to address the problem at the early phase of DNA computation is preferred. The proposed genetic code based approach targets early coding stage of DNA computation.

B. Genetic code is highly error resistant

The genetic code is a triplet code based on three-letter codons. The complete genetic code is shown in Fig. 2, where the 64 triplets stand for one or another of the 20 amino acids, and the stop codons. ATG coded for Methionine is the start codon. Three of the codons, TAA, TAG, and TGA, are stop codons. Clearly, a given amino acid may be encoded by more than one codon, but a codon can code for only one amino acid. It is called redundancy. The redundancy is not evenly divided among amino acids. For example, Methionine and Tryptophan are represented by only one codon each, where Leucine, Arginine and Serine are represented by six different codons. Mathematically, the triplet codon space can be regarded as

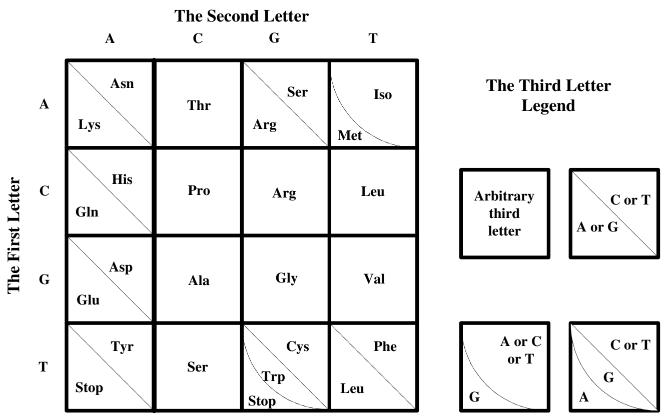


Fig. 2. Table of the genetic code. Each codon consists of three letters. For all codons, the first and the second letters are important to define what the codon represents for. For those squares without diagonal lines, it means that the third letter does not matter. If a square has a diagonal line, the upper right codon is represented by choosing the third letter as either T or C, and the lower left codon can be represented with the third letter as either A or G.

the binary cartesian product or cartesian square on the set $\{$A,T,G,C$\}$. As a nature evolution of life, we believe that the 20 amino acids are the result of optimization. Chemical structures of these molecules are unique for encoding information.

The genetic code provides the specificity for protein synthesis. The genetic information in an mRNA molecule can be regarded as a series of non-overlapping three-letter "words." Each sequence of the three nucleotides along the chain specifies a particular amino acid. Each codon is complementary to the corresponding triplet in the DNA molecule from which it is transcribed [12].

The genetic code has great quality assurance, because of its redundancy. Some degree of mis-binding results in no change in the coding words. These mis-binded bases are called silent or synonymous errors, which are not expressed in protein expression. Similar to silence mutations in cells, the silent or synonymous errors are quite common, and lead to genetic diversity that is not expressed as phenotype differences.

From a mathematical point of view, the genetic coding system is an optimal solution for encoding communication signals. Since there are only four base letters (A,T,G,C), a one-letter code clearly cannot unambiguously encode 20 amino acids. A two-letter code could only define $4 \times 4 = 16$ codons – still not enough. But a triplet code could encode up to $4 \times 4 \times 4 = 64$ codons. It seems that it has enough redundancies for errors during transcriptions and translation. It is not necessary for four-letter codes, which could include $4 \times 4 \times 4 \times 4 = 256$ codons for 20 amino acids.

If a DNA computation problem is coded using redundant genetic codes, the code will be highly error resistant and the error rates will be low. Different from biological systems that are highly diverging and require a large set of DNA codons to code various genetic information, the set of "codons" for DNA computation will be small. To take advantage of chemical properties of the molecular structures and obtain a reduced set of "codons" (called coding set for DNA computation), we will reduce the regular codons of biological systems to

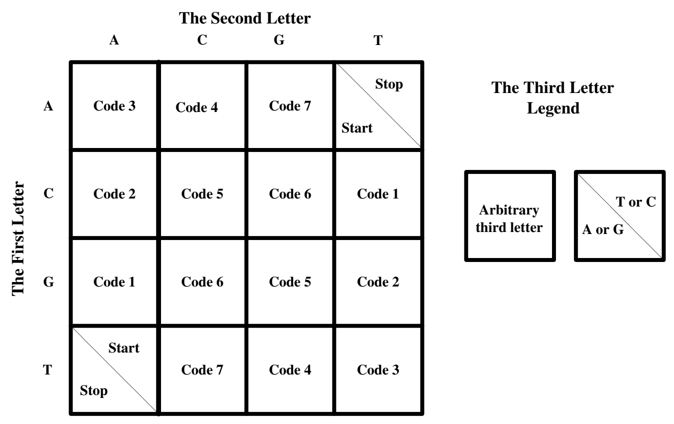


Fig. 3. Coding set for DNA computation

obtain smaller coding sets. One additional concern is that all coding sets should be closed, i.e., elements of a coding set and their complementaries (anti-codon) are within the same coding set. This is based on the concern that DNA computation does not use the same mechanism as biological systems for recognizing DNA strands. We expect to have a robust coding system. Second, the triple code mechanism will still be used. Since there may be biochemistry and stability reasons for the triple codes, though it is not completely clear at the present time.

Based on the above discussions, the following coding sets are obtained as shown in Fig. 3.

- Start coding set: ATG, ATA, TAC, or TAT.
- Coding set 1: GAA, GAG, GAT, GAC, CTT, CTC, CTA, or CTG.
- Coding set 2: CAA, CAG, CAT, CAC, GTT, GTC, GTA, or GTG.
- Coding set 3: AAA, AAG, AAC, AAT, TTT, TTC, TTG, or TTA.
- Coding set 4: ACC, ACA, ACG, ACT, TGG, TGT, TGC, or TGA.
- Coding set 5: CCT, CCC, CCA, CCG, GGA, GGG, GGT, or GGC.
- Coding set 6: GCT, GCC, GCA, GCG, CGA, CGG, CGT, or CGC.
- Coding set 7: TCT, TCC, TCA, TCG, AGT, AGC, AGA, or AGG.
- Stop coding set: TAA, TAG, ATT, or ATC.

It can be easily verified that all the above coding sets are closed with respect to the DNA complementary operation. In addition, the above coding scheme allows significant amount of overlapping, which leads to high error resistance. The coding sets are enough to code significantly large problems by varying the length of wording. By coding this way, many DNA base mutations may not cause changes in word meaning for DNA computation.

C. Genetic Code Based DNA Computation to Solve the Hamiltonian Path Problem (HPP)

The HPP problem is to find (if there is) a Hamiltonian path for a given graph. A Hamiltonian path is a sequence of compatible one-way edges of a directed graph that begins and ends at a specified vertex and enters every other vertex exactly once. Known algorithms for this problem have exponential worst-case complexity. The problem has been proved to be NP-complete.

Assume that the graph has $n > 0$ vertices (cities) and i is the index of a vertex [2]. The following steps are usually followed.

1) Associate the start vertex $i = 1$ with one code (three bases) from the start coding set. Associate the end vertex $i = n$ with a code from the stop coding set. For the convenience of later sequence extraction, it is preferred to have all edges coded with equal length. So only the first three DNA bases are needed for the start and end cities. To match the sequence length requirements for each vertex, fill in the rest sequences with random DNA bases.
2) Associate each of the other vertices i $(1 < i < n)$ with a $3m$-mer sequence generated by m codes (usually an even number to keep the left and right side edges of a city with equal length), and denote it by O_i. For each edge $i \rightarrow j$, an oligonucleotide $O_{i \rightarrow j}$ is created, which is the 3' $3m/2$-mer of O_i followed by the 5' $3m/2$-mer of O_j. This construction process preserves edge orientation. The $\bar{O}_i$ serves as splints to bring oligonucleotides associated with compatible edges together for ligation.
3) Keep only those paths that begin with codes from the start coding set, and end with codes from the stop coding set. This can be done by PCR amplifying products of the Step 1) using primers starting with codes from the start coding set or the stop coding set. Thus only those molecules encoding paths, which begin with vertex 1 and end with vertex n, are amplified.
4) Keep only those paths that enter exactly n vertices. The product of Step 2) is run on an agarose gel and the $3m$ base pair band (corresponding to dsDNA encoding paths entering exactly n vertexes) is excised and soaked to extract DNA.
5) Keep only those paths that enter all vertices of the graph exactly once. To achieve this by affinity purify the product of Step 4) using a biotin-avidin magnetic beads system. This can be done by first generating single-stranded DNA sequences from the dsDNA product of Step 4) and then incubating the ssDNA with $\bar{O}_2$ conjugated to magnetic beads. Only those ssDNA molecules containing O_2 (and hence encoded paths which enter vertex 2 at least once) anneal to the bound $\bar{O}_2$ and were retained. The process repeat successively with $\bar{O}_3$, $\bar{O}_4$, ..., $\bar{O}_{n-1}$ and $\bar{O}_n$.
6) The remaining DNA sequences (paths) in the test tube represent solutions.

To further illustrate the idea, consider an $n = 7$ vertex HPP graph as given in [2], we use 12 DNA bases (4 codes from the coding sets) to uniquely code each of the 7 cities. The following is one possibility.

- City 1: code the start city with ATG plus three additional

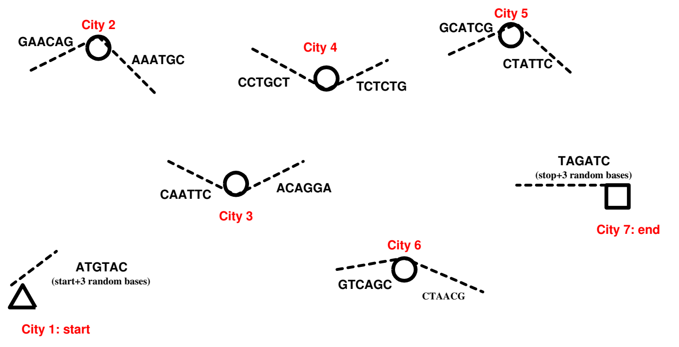


Fig. 4. Coding for the seven cities of the HPP problem

DNA bases to match the length requirements. Only the first three bases are used to identify the city. We choose ATGTAC here.

- City 2: one edge is coded by GAACAG. Theoretically, any pair of codes from the coding set 1 and the coding set 2 can be used. The other edge is coded by any pair from the coding set 3 and the coding set 4. AAATGC is used here.
- City 3: one edge is coded by any pair of codes from the coding set 2 and the coding set 3. We choose CAATTC. The other edge is coded by any pair from the coding set 4 and the coding set 5, such as ACAGGA.
- City 4: one edge is coded by any pair of codes from a pair of the coding set 5 and the coding set 6. CCTGCT is used here. The other edge is coded by any pair from the coding set 7 and the coding set 1, such as TCTCTG.
- City 5: one edge is coded by any pair of codes from the coding set 6 and the coding set 7. GCATCG is used here. The other edge is coded by any pair of codes from the coding set 1 and the coding set 3, such as CTATTC.
- City 6: one edge is coded by any pair of codes from the coding set 2 and the coding set 7. GTCAGC is chosen here. The other edge is coded by any pair from the coding set 1 and the coding set 4, such as CTAACG.
- City 7: code the end city with TAG, plus three random DNA bases to match the length requirement. Only the first three DNA bases will be used to identify the city. We choose TAGATC.

Fig. 4 shows the final coding graph for each edge and vertex. The next step is to apply DNA molecular techniques to obtain biological solutions. The above coding approach does not affect any of these operations. Similar to the work in [2], same conclusion will be obtained. Compared with the brute-force approach used for solving HPP problem in [2], the proposed approach can save 40% of the coding characters, which will eventually speed up the problem solving time. More importantly, the proposed approach has a much lower error rate.

D. Computation Formulation

Through the above genetic coding, it is expected that the error rates can be reduced. To demonstrate the idea, assume that the following stochastic transfer matrix of DNA sequences are held for DNA bases $\{A, T, G, C\}$.

$$\Gamma = \begin{bmatrix} 0.9990 & 0.0003 & 0.0004 & 0.0003 \\ 0.0003 & 0.9990 & 0.0003 & 0.0004 \\ 0.0004 & 0.0003 & 0.9990 & 0.0003 \\ 0.0003 & 0.0004 & 0.0003 & 0.9990 \end{bmatrix} \tag{11}$$

After one transformation, a DNA sequence $X = GATCAG$ coded by codons from the coding sets 1 and 2 can be expressed in numerical values as

$$\Gamma_1 = \begin{bmatrix} 0.0004 & 0.0003 & 0.9990 & 0.0003 \\ 0.9990 & 0.0003 & 0.0004 & 0.0003 \\ 0.0003 & 0.9990 & 0.0003 & 0.0004 \\ 0.0003 & 0.0004 & 0.0003 & 0.9990 \\ 0.9990 & 0.0003 & 0.0004 & 0.0003 \\ 0.0004 & 0.0003 & 0.9990 & 0.0003 \end{bmatrix} \tag{12}$$

where each element shows the probability that the corresponding base is obtained after molecular manipulation.

It can be concluded that the original sequence $X = GATCAG$ is still preserved very well. After about 1000 transformations, the sequence turns to be ambiguous as follows.

$$\Gamma_2 = \begin{bmatrix} 0.2021 & 0.1748 & 0.4484 & 0.1748 \\ 0.4484 & 0.1748 & 0.2021 & 0.1748 \\ 0.1748 & 0.4484 & 0.1748 & 0.2021 \\ 0.1748 & 0.2021 & 0.1748 & 0.4484 \\ 0.4484 & 0.1748 & 0.2021 & 0.1748 \\ 0.2021 & 0.1748 & 0.4484 & 0.1748 \end{bmatrix} \tag{13}$$

It seems that the original sequence may be turned into a different format. However, by following the proposed coding scheme, the original word coded in X is still well preserved. This is because of the redundancy of the coding scheme. There are $8 \times 8 = 64$ different combinations of codings for the six-base DNA sequence X, and they are all coded for the same word. Even mutation occurs after multiple transformations, the original information remains intact. This is the beauty of the genetic coding based method. However, if a brute force fixed length coding approach is used, i.e., each combination represents one scheme only, the original coding meaning cannot still be kept intact. If any other redundancy coding approach that has less than 64 combinations to represent one coding scheme is used, the proposed genetic coding scheme still performs the best.

Similarly, the HPP can be programmed and described at the test tube level as:

- Input $T_i(X)$, $i \in N$, which contains DNA segments encoding all the cities.
- Amplification: $\times T_i$, $(i \in N)$ generate new large sets of DNA sequences representing all different cities.
- Mergence: mix $T_1 \cup T_2$ to generate new test tube T. The process finds connections between different cities.
- Separation: $-(T, X)$ separating strand X from the test tube T, where X is a DNA strand of length 84 and connecting edges of all cities.
- Detection: $T?X$ to find final DNA strand X representing pathes connecting all cities exact once.

The problem can be programmed and described at DNA strand level as:

- For a city coded by $X_iY_i, i \in N$, hybridization may lead to $(X_i\bar{X}_i)(Y_i\bar{Y}_i)$. The process can be described as $X_iY_i \bullet \bar{X}_i\bar{Y}_i \rightarrow (X_i\bar{X}_i)(Y_i\bar{Y}_i)$.
- Ligation may happen during the process. $\bar{Y}_{i-1}(X_i\bar{X}_i)(Y_i\bar{Y}_i)\bar{Y}_{i+1} \quad + \quad X_{i-1}Y_{i-1} \quad \rightarrow [X_{i-1}(Y_{i-1}\bar{Y}_{i-1})(X_i\bar{X}_i)(Y_i\bar{Y}_i)\bar{Y}_{i+1}]$.

IV. Applications

The above mathematical formulation of DNA computation may be used in the following applications.

A. Word design for DNA computation

In DNA computation, to reliably store and retrieve information in synthetic DNA strands, DNA word design is very important. DNA word design is to design sets of equal-lengthed words over the alphabet $\{A, T, G, C\}$ satisfying certain constraints. The primary constraint is $\forall X, Y \in \{$DNA words$\}$. There are at least d mismatches between X and Y, and between X^R and $\bar{Y}$, where X^R represents the reverse of X [11].

Based on the above formulation, the word design problem is equivalent to the following mathematical problem: choosing X and Y sequences from characters $\{A, T, G, C\}$, so that the number of non-zero bases from the base-by-base operation of $f(X)$ plus $f(Y)$, and $f(X^R)$ plus $f(\bar{Y})$ are greater or equal to d. Based on the above discussion, the solvability of the problem and the upper and lower bounds of DNA word design can be answered [6].

B. Natural DNA processing

The above formulation may also be used to process natural DNA for sequencing, fingerprinting and mutation detection. The idea is to first convert character-based DNA sequences into numerical sequences, then apply numerical computation techniques.

An example is multiple DNA sequence alignment. The proposed mathematical formulation may save significant amount of machine time, if the sequences are expressed and compared in the numerical domain.

For example, to detect long DNA sequence mutations as shown in equation (14), the proposed method can be applied first to convert the character-based sequences into numerical sequences as shown in (15). A numerical minus operation can then be conducted. If the final result is non-zero, this implies that there is mutation. The comparison is efficient by avoiding tedious character-based side-by-side comparison. This is called re-coding DNA. As pointed in [13], the re-coding has great potential application for DNA engineering applications. The formulation proposed in this paper provides an ideal mechanism for the re-coding process.

$$\begin{array}{c} ATTCCAGA \cdots GACCTTGAGT \\ ATTCCATA \cdots GACCTCGAGT \end{array} \tag{14}$$

$$\begin{array}{c} 03311020 \cdots 2011332023 \\ 03311030 \cdots 2011312023 \end{array} \tag{15}$$

where "$\cdots$" may represent thousands or millions of DNA bases.

C. Combinatorial chemistry

The mathematical formulation may also be used in combinatorial chemistry for pseudo-enzyme design [4]. The goal is to create molecules with desired properties, which may be difficult or expensive by direct experimental studies. With the above mathematical formulation, the molecule design process can be done in a numerical simulation mode. The beauty of this approach is that the tedious and expensive pseudo-molecule experimental design process may be avoided.

It is also interesting to note that most advanced numerical operations for conventional computers require combinations of a number of basic silicon computer operations. However, they may be completed by much less operations using DNA computation as shown in the mathematical operations. The above symbolic operation may be used for algorithm implementation in DNA computation.

V. Discussion and Conclusions

A mathematical formulation of DNA computation has been proposed in this paper. Propositions related to the formulation have also been presented. Even though DNA computation is still an open problem for practical implementation, numerous studies have suggested that it is an interesting field for fast computation of large NP-complete problems.

Currently, the high error rate is a major concern for DNA computation. This paper proposes a genetic code based DNA computation method to reduce the error rate. The idea is inspired by the genetic code of biological systems, except that the codon sets have been reduced for DNA computation. Redundancy in the genetic codes plays an important role in reducing error rates. Different from many methods proposed in the open literature [6], the proposed method does not require any additional bio-molecular techniques or steps, and it is not limited to any specific type of errors, which is a great advantage compared with other methods in the open literature.

In addition, the formulation can easily convert a character-based DNA computation problem into a numerical value based computing problem. This will allow researchers to build theoretical framework for DNA computation, and analyze algorithmic as well as computational efficiency of DNA computation. Some potential problems may be further studied within the proposed mathematical framework, such as how can we increase DNA computational efficiency? what is the ultimate limit of the error-rate for DNA computation using genetic coding? what are the solvability and the upper or lower bounds for the DNA word design?

We may also easily conclude from the formulation that DNA computation is very fast. This is partially due to many fundamental operations, such as plus, minus, multiplication, division, and set operations are single step operations, which are much faster than conventional silicon computation.

The goal of this paper is to start a discussion on mathematical formulations of DNA computation. The formulation may be further used to understand the theoretical aspects of DNA computation, and be used for natural DNA processing as well as pseudo-enzyme design for combinatorial chemistry. The proposed genetic code based DNA computation demonstrates promising potential to reduce high reduce error rates. However, this paper is in no way a complete mathematical subject on DNA computation. We hope more discussions can be inspired in that regard.

Acknowledgment

The authors would like to thank the reviewers' constructive comments and excellent suggestions to bring the paper into current format.

References

[1] J. Adams. On the Application of DNA Based Computation. *http://publish.uwo.ca/ jadams/dnaapps1.htm*, 1998.

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

[3] L. Adleman, On Constructing A Molecular Computer, *DNA Based Computers, Eds. R. Lipton and E. Baum, DIMACS: series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society*, pp. 1-21, 1996.

[4] D. Bartel, and J. Szostak, Isolation of new ribozymes from a large pool of random sequences. *Science*, vol. 261, pp. 1411-1418, September 1991.

[5] E. Baum, and D. Boneh, Running dynamic programming algorithms on a DNA computer. *the 2nd DIMACS workshop on DNA based computers*, pp. 141-147, 1996.

[6] D. Boneh, and R. Lipton, Making DNA computers error resistant. *Technical report, Princetion University*, CS-TR-491-495, 1996.

[7] R. Durrett, Probability models for DNA sequence evolution. *New York : Springer*, 2002.

[8] T. Eng, and B. Serridge, A Surface-Based DNA Algorithm for Minimal Set Cover. *the 3rd DIMACS workshop on DNA based computers*, pp. 74-82, June 1997.

[9] R. Lipton, Using DNA to solve NP-Complete Problem. *Science*, vol. 268, pp. 542-545, April 1995.

[10] C. C. Maley, DNA computation: theory, practice, and prospects, *Evolutionary computation*, vol. 6, no. 3, pp. 201-229, 1998.

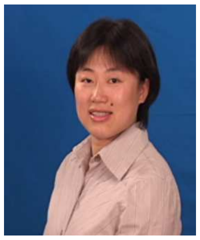
[11] A. Marathe, A. E. Condon, and R. M. Corn, On combinatorial DNA word design, *J. Computational Biology*, vol. 8, no. 3, pp. 201-220, 2001.

[12] W. K. Purves, D. Sadava, G. H. Orians and H. C. Heller. Life: the science of biology, *Sinauer Associates, Inc.*, 6th edition, 1994.

[13] J. H. Reif. Paradigms for Biomolecular Computation, *Unconventional Models of Computation*, edited by C. S. Calude, J. Casti, and M. J. Dinneen, Springer-Verlag, New York, January 1998, pp 72-93.

[14] P. W. K. Rothemund. A DNA and restriction enzyme implementation of Turing machines. In R. J. Lipton & E. B. Baum (Eds.), DNA Based Computers, pp. 1-22, Providence, RI: American Mathematical Society, 1996.

Mingjun Zhang (S'98-M'01) received the D.Sc. degree from Washington University in St. Louis, USA in 2000. Since 2001, he has been with the Life Sciences and Chemical Analysis Division of Agilent Technologies, USA. He was awarded the Early Career Award (Industry) by the IEEE Robotics and Automation Society in 2003. Currently, he serves as an Associate Editor for the IEEE Transactions on Automation Science and Engineering, and is on the Editorial Board of the Journal of Nanomedicine: Nanotechnology, Biology and Medicine. His research interests include DNA nanotechnology, micro-/nano-scale system dynamics and control, and automation for the life sciences.



Maggie Cheng (M'03) received her Ph.D in Computer Science and Engineering from the University of Minnesota at the Twin Cities in 2003. She is currently an assistant professor at the Computer Science Department in the University of Missouri, Rolla. Her current research interests include wireless networking and mobile computing, sensor networks, network security, theory of computing, and combinatorial optimization. She is a member of the IEEE.

Tzyh-Jong Tarn (M'71-M'83-F'85) received the D.Sc. degree in control system engineering from Washington University in St. Louis, USA. Currently, he is a Professor in the Department of Electrical and Systems Engineering and the Director of the Center for Robotics and Automation at Washington University. He served as the President of the IEEE Robotics and Automation Society from 1992 to 1993, the Director of the IEEE Division X, from 1995 to 1996, and was a member of the IEEE Board of Directors, from 1995 to 1996. At present, he serves as the Vice President for Conferences of the IEEE Robotics and Automation Society. He received the NASA Certificate of Recognition for the creative development of a technical innovation on "Robot Arm Dynamic Control by Computer" in 1987. The Japan Foundation for the Promotion of Advanced Automation Technology presented him with the Best Research Article Award in March 1994. He also received the Best Paper Award at the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. He is the first recipient of both the Nakamura Prize and the Ford Motor Company best paper award at the Japan/USA Symposium on Flexible Automation in 1998. In addition, he is the recipient of the prestigious Joseph F. Engelberger Award of the Robotic Industries Association in 1999, the Auto Soft Lifetime Achievement Award in 2000 and the Pioneer in Robotics and Automation Award in 2003 from the IEEE Robotics and Automation Society. He was featured in the Special Report on Engineering of the 1998 Best Graduate School issue of US News and World Report and his recent research accomplishments were reported in the "Washington Times", Washington D.C., the "Financial Times", London, "Le Monde", Paris, and the "Chicago Sun-Times", Chicago, etc.