Procedures for a dynamical system on \(\{0, 1\}^n\) with DNA molecules

Dongmei Xiaoa, Wenxia Lib, Jiang Yua, Xiaodong Zhanga, Zhizhou Zhangc, Lin Hec

\(a\) Department of Mathematics, Shanghai Jiao Tong University, Shanghai 200030, PR China
\(b\) Department of Mathematics, East China Normal University, Shanghai 200062, PR China
\(c\) Bio-X DNA Computer Consortium, Shanghai Jiao Tong University, Shanghai 200030, PR China

Received 14 April 2005; received in revised form 9 November 2005; accepted 23 November 2005

Abstract

In this paper, an improved form of DNA representations of elements in \(\{0, 1\}^n\), which was first proposed by Fujiwara et al. [Fujiwara, A., Matsumoto, K., Chen, W., 2004. Procedures for logic and arithmetic operations with DNA molecules. Int. J. Found. Comput. Sci. 15, 461–474], is given. Using this improved representations, a procedure for cycling shift is proposed, and this procedure can be implemented in \(O(1)\) lab steps theoretically. Based on the operation for cycling shift, dynamic behavior of an operator on \(\{0, 1\}^n\) is investigated by DNA molecules.

© 2005 Elsevier Ireland Ltd. All rights reserved.

Keywords: DNA representation; DNA computing; Symbolic space; Dynamic behavior

1. Introduction

A deoxyribonucleic acid (DNA) is a polymer, which can be strung together from monomers called deoxyribonucleotides (Păun et al., 1998). Distinct nucleotides are detected only with their bases. Those bases are, respectively, abbreviated as A (adenine), G (guanine), C (cytosine) and T (thymine). Under appropriate conditions two strands of DNA can form a double strand, if the respective bases are the Watson–Crick complements of each other—A matches T and C matches G; also the 3'-end matches the 5'-end, e.g., the single strands 5'-ACCTGGATGTAA-3' and 3'-TGGACCTACATT-5' can form a double strand. We also call the strand 3'-TGGACCTACATT-5' as the complementary strand of 5'-ACCTGGATGTAA-3' and simply denote it by ACCTGGATGTAA.

DNA has two important features, which are Watson–Crick complementarity and massive parallelism. As the first work for DNA computing, Adleman (1994) took advantage of these features to present an idea of solving the Hamiltonian path problem (an NP problem) of size \(n\) in \(O(n)\) steps using DNA molecules. Lipton (1995) demonstrated that Adleman’s experiment could be used to determine the NP-complete satisfiability (SAT) problem (the first NP-complete problem). Ouyang et al. (1997) presented a molecule biology-based experimental solution to the maximal clique NP-complete problem. Each of these works are based on performing some basic operations on DNA strands.
A (test) tube is a set of molecules of DNA, i.e., a multi-set of finite strings over the alphabet \{A, C, G, T\}. Given a tube, one can perform the following basic operations:

1. **Merge**: Given two test tubes \(T_1, T_2\), \(\text{Merge}(T_1, T_2)\) stores the union \(T_1 \cup T_2\) in \(T_1\).
2. **Copy**: Given a test tube \(T_1\), \(\text{Copy}(T_1)\) produces a test tube \(T_2\) with the same contents as \(T_1\).
3. **Detect**: Given a test tube \(T\), \(\text{Detect}(T)\) outputs “yes” if \(T\) contains at least one strand, otherwise, \(\text{Detect}(T)\) outputs “no”.
4. **Separation**: Given a test tube \(T_1\) and a set of strings \(X\), \(\text{Separation}(T_1, X)\) removes all single strands containing a string in \(X\) from \(T_1\) and produces a test tube \(T_2\) with the removed strands.
5. **Discard**: Given a tube \(T\), \(\text{Discard}(T)\) will discard the tube \(T\).
6. **Cleavage**: Given a test tube \(T\) and a string of two symbols \(\sigma\sigma_0\), \(\text{Cleavage}(T, \sigma\sigma_1)\) cuts each double strand containing \(\sigma\sigma_1\) in \(T\) into two double strands as follows:
   \[
   \sigma\sigma_1 \rightarrow \sigma\sigma_0, \sigma_0\sigma_1
   \]
7. **Annealing**: Given a test tube \(T\), \(\text{Annealing}(T)\) produces all feasible double strands in \(T\). (The produced double strands are still stored in \(T\) after Annealing.)
8. **Denaturation**: Given a test tube \(T\), \(\text{Denaturation}(T)\) dissociates each double strand in \(T\) into two single strands.
9. **Selection**: Given a test tube \(T_1\) and an integer \(L\), \(\text{Selection}(T_1, L, T_2)\) removes all strands, whose length is \(L\), from \(T_1\), and produces a test tube \(T_2\) with the removed strands.
10. **Append**: Given a tube \(T\) and a short DNA single strand \(Z\), \(\text{Append}(T, Z)\) will append \(Z\) onto the end of every strand in the tube \(T\).
11. **Read**: Given a tube \(T\), the operation \(\text{Read}(T)\) is used to describe a single molecule, which is contained in the tube \(T\). Even if \(T\) contains many different molecules each encoding a different set of bases, the operation can give an explicit description of exactly one of them.

Since these eleven manipulations are implemented with a constant number of biological steps for DNA strands (Plum et al., 1998), we assume that the complexity of each manipulation is \(O(1)\) steps.

In order to apply DNA computing on a wide range of problems, procedures for primitive operations, such as logic or arithmetic operations, are needed. There are some works for primitive operations in DNA computing (Frisco, 2002; Guarnieri et al., 1996; Gupta et al., 1997; Hug and Schuler, 2001). As we know, the key to solve a mathematical problem using DNA molecules is to design appropriate DNA representations according to the problem itself. A canonical DNA representation for a binary number was introduced by Fujiwara et al. (2004). It deals with DNA representations of a binary number of \(m\) bits. A value of the \(i\)th binary bit of the \(j\)th binary number is represented by a single strand \(S_{i,j}\) such that

\[
S_{i,j} = E_1 N_i B_i C_0 C_j V_{i,j} E_0,
\]

where \(C_0, C_1, E_0, E_1, B_j, V_{i,j}\) and \(N(1 \leq i \leq m, 1 \leq j \leq n)\) all are single strands. \(V_{i,j}\) takes the single strands \(0\) or \(1\), representing the real numbers zero or one, respectively. Single strands \(C_0, C_1\) and \(E_0, E_1\) are special symbols cut by Cleavage. For example, the following sets of single strands:

\[
\{E_1 N_1 B_1 C_0 C_1 V_{1,0} E_0, E_1 N_1 B_1 C_0 C_1 E_0, E_1 N_1 B_1 C_0 C_1 0 E_0\},
\]

and

\[
\{E_1 N_2 B_1 C_0 C_1 V_{2,0} E_0, E_1 N_2 B_1 C_0 C_1 0 E_0, E_1 N_2 B_1 C_0 C_1 0 E_0\}
\]
denote \(2 (m = 2)\) binary numbers of \(3 (n = 3)\) bits: 110 and 100, respectively. Based on the representation by (1), Fujiwara et al. (2004) design a very important operation, denoted by ValueAssignment(\(T_{\text{input}}, T_{\text{output}}\)), which assigns
the same value \( V \in \{0, 1\} \) to each bit in the input tube \( T_{\text{input}} \) and can be implemented in \( O(1) \) biological steps. More exactly, if
\[
T_{\text{input}} = \{ E_i N_i B_i C_i V_i E_0 : 1 \leq i \leq m, 1 \leq j \leq n \},
\]
where \( V_{i,j} \in \{0, 1\} \), then ValueAssignment\( _{0} (T_{\text{input}}) \) gives that
\[
T_{\text{input}} = \{ E_i N_i B_i C_i V_i E_0 : 1 \leq i \leq m, 1 \leq j \leq n \}.
\]
By means of the operation of ValueAssignment, Fujiwara et al. (2004) proposed procedures for logic and arithmetic operations with DNA molecules. Since each element of \( \{0, 1\}^n \) can be considered a binary number of \( n \) bits, the above representation and the operation of ValueAssignment can be used for some matters on \( \{0, 1\}^n \), e.g., permutation of the first two terms of an element, i.e., change \((V_1, V_2, V_3, \ldots, V_n)\) into \((V_2, V_1, V_3, \ldots, V_n)\) for \((V_1, V_2, V_3, \ldots, V_n) \in \{0, 1\}^n\). However, a typical operation, the cycling shift which changes \((V_1, V_2, V_3, \ldots, V_n)\) into \((V_2, V_1, V_3, \ldots, V_n)\) for \((V_1, V_2, V_3, \ldots, V_n) \in \{0, 1\}^n\), needs to implement the operation of ValueAssignment \( n - 1 \) times. To remedy this shortage, we improve the representation in (1) by setting
\[
S_{i,j} = E_i N_i A_j D_0 B_j C_j V_i E_0,
\]
and propose a procedure for the operation of CycleShift. In Section 3, a procedure is proposed for investigating the orbit of ValueAssignment and CycleShift.


2. Bit representation and cycling transformation

Let \( \Sigma \) be a set of single strands such that
\[
\Sigma = \{ A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_n, C_0, C_1, C_2, D_0, D_1, E_0, E_1, 1, 0, \#, \bar{1}, \bar{0} \}.
\]
Both \( A_i \) and \( B_i \), \( i = 1, 2, \ldots, n \) are used to denote the \( i \)th bit position for an element (or a \( 0 - 1 \) sequence) of \( \{0, 1\}^n \).
and Cleavage($T, E_0, E_1$) cut all double strands containing
\[
\begin{bmatrix}
C_0C_1 \\
C_2C_3
\end{bmatrix}, \quad
\begin{bmatrix}
D_0D_1 \\
D_2D_3
\end{bmatrix}
\quad \text{and}
\begin{bmatrix}
E_0E_1 \\
E_2E_3
\end{bmatrix}
\]
in a test tube $T$, respectively. Symbols "0" and "1" are used to denote values of bits, and $\theta$ is a special symbol for Separation.

Let $([V_1, V_2, \ldots, V_n] \in \{0, 1\}^n).$ Using the above notations, a value of bit at position $j$ is represented by a single strand $S_j,$ called as a memory strand, such that
\[
S_j = E_jA_jD_jB_jC_jC_1V_0E_0,
\]
where $V_j = 0$ if a value of the bit is 0, otherwise, $V_j = 1.$ For instance, the representation of $(1, 0, 0) \in \{0, 1\}^3$ is
\[
\begin{bmatrix}
E_1A_1D_1B_1C_10C_1V_0E_0, E_2A_2D_2B_2C_200E_0, E_3A_3D_3B_3C_300E_0
\end{bmatrix}.
\]

Compared with that in (2), for simplicity we omit $N_i$ and only use one index $j$ in (4) since we do not consider multiple elements of $\{0, 1\}^n$ simultaneously.

In the following we show a procedure for the operation CycleShift. It is implemented in $O(1)$ lab steps. Let
\[
(V_1, V_2, \ldots, V_n) \in \{0, 1\}^n.
\]
The CycleShift transfers $(V_1, V_2, \ldots, V_n)$ into $(V_2, \ldots, V_n, V_1).$ More exactly, if
\[
T_{\text{input}} = \begin{bmatrix}
E_1A_1D_1B_1C_1V_1E_0 \end{bmatrix} \quad \begin{bmatrix} 1 \leq i \leq n \end{bmatrix},
\]
then after performing CycleShift($T_{\text{input}}$, $T_{\text{output}}$) we have
\[
T_{\text{output}} = \begin{bmatrix}
E_1A_1D_1B_1C_1V_0E_0, E_1A_1D_1B_1C_1V_1E_0n \end{bmatrix} \quad \begin{bmatrix} 1 \leq i \leq n - 1 \end{bmatrix}.
\]

Some auxiliary test tubes are given by
\[
T_p = \begin{bmatrix}
D_1D_2
\end{bmatrix}, \quad T_c = \begin{bmatrix}
C_0C_1
\end{bmatrix}, \quad T_k = \begin{bmatrix}
E_1A_1A_2D_1A_2A_3D_2B_1C_1C_0E_0
\end{bmatrix}, i = 1, \ldots, n - 1.
\]
and
\[
T_b = \begin{bmatrix}
D_1B_1C_0, E_1A_1D_1B_1C_1V_0, E_0 \end{bmatrix},
\]
where $V_0 \in \{0, 1\}$, then after performing CycleShift($T_{\text{input}}$, $T_{\text{output}}$) is implemented in five steps. For reader's convenience, we show the contents in the tubes after some operations.

**Procedure** CycleShift($T_{\text{input}}$, $T_{\text{output}}$)

Step 1: Shift subscripts of $A$’s

Separation($T_{\text{input}}$,$\{A_1\}$,$\{A_2\}$,$\{A_3\}$)

\[ \Rightarrow \quad T_{\text{input}} = \begin{bmatrix}
E_1A_1D_1B_1C_1V_0E_0
\end{bmatrix}. \]

Separation($T_{\text{input}}$,$\{A_1\}$,$\{A_2\}$,$\{A_3\}$)

\[ \Rightarrow \quad T_{\text{input}} = \begin{bmatrix}
E_1A_1D_1B_1C_1V_0E_0
\end{bmatrix}. \]

Copy($T_{\text{input}}$,$T_{\text{output}}$)

Merger($T_{\text{input}}$,$T_{\text{output}}$)

\[ \Rightarrow \quad T_{\text{output}} = \begin{bmatrix}
E_1A_1D_1D_2B_1C_1V_0E_0, 2 \leq i \leq n
\end{bmatrix}. \]

Copy($T_{\text{input}}$,$T_{\text{output}}$)

Merger($T_{\text{input}}$,$T_{\text{output}}$)

\[ \Rightarrow \quad T_{\text{output}} = \begin{bmatrix}
E_1D_1B_1C_0, E_0 \end{bmatrix}, 2 \leq i \leq n. \]

Annealing($T_{\text{input}}$)

\[ \Rightarrow \quad T_{\text{input}} = \begin{bmatrix}
E_1A_1D_2B_0C_0V_0E_0, 2 \leq i \leq n
\end{bmatrix}. \]

Cleavage($T_{\text{input}}$,$D_1$,$D_0$)

\[ \Rightarrow \quad T_{\text{input}} = \begin{bmatrix}
E_1A_1D_0, D_1B_0C_0V_0E_0, 2 \leq i \leq n
\end{bmatrix}. \]
Denaturation ($T_{\text{input}}$)

$T_{\text{input}} = \{D_0, D_1, E_1 A_i D_0, D_1 B_i C_i V_i E_0 | 2 \leq i \leq n\}$. 

Separation($T_{\text{input}}$) ($C_1 \setminus C$), $T_{\text{output}}$

$T_{\text{input}} = \{D_0, D_1, E_1 A_i D_0, 2 \leq i \leq n\}$. 

$T_{\text{output}} = \{D_1 B_i C_i V_i E_0 | 2 \leq i \leq n\}$. 

Copy($T_{\text{input}}$, $T_{\text{copy}}$)

$T_{\text{output}} = \{E_1 A_{i-1} D_0, E_1 A_i D_0 | 2 \leq i \leq n\}$. 

Annealing($T_{\text{output}}$)

$T_{\text{output}} = \left\{ \begin{array}{l} E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 \\
E_1 A_i D_0 D_1 B_i C_i V_i E_0 \\
\end{array} \right\} \forall i \leq n$. 

Denaturation($T_{\text{output}}$)

$T_{\text{output}} = \{E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 | 2 \leq i \leq n\}$. 

Separation($T_{\text{output}}$) ($D_0(D)\setminus T$, $T_{\text{tmp}}$)

$T_{\text{output}} = \{E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 | 2 \leq i \leq n\}$. 

$T_{\text{tmp}} = \{E_1 A_i D_0 D_1 B_i C_i V_i E_0 | 2 \leq i \leq n\}$. 

Discard($T_{\text{tmp}}$)

Separation($T_{\text{input}}$, $\{0\}$, $T_{\text{to}}$)

$T_{\text{to}} = \{E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 | V_i = 0, 2 \leq i \leq n\}$. 

Separation($T_{\text{input}}$, $\{1\}$, $T_{\text{to}}$)

$T_{\text{to}} = \{E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 | V_i = 1, 2 \leq i \leq n\}$. 

Step 2: Shift subscripts of $B$'s for single strands in tube $T_0$

Copy($T_{\text{to}}$, $T_{\text{copy}}$)

$T_{\text{output}} = \{C_0 C_1, E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 | V_i = 0, 2 \leq i \leq n\}$. 

Annealing($T_{\text{output}}$)

$T_{\text{output}} = \left\{ \begin{array}{l} E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 \\
C_0 C_1 \end{array} \right\} \forall i \leq n$. 

Cleavage($T_{\text{output}}$, $C_0 C_1$)

$T_{\text{output}} = \left\{ \begin{array}{l} E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 \\
C_0 C_1 V_i E_0 \\
\end{array} \right\} \forall i \leq n$. 

Denaturation($T_{\text{output}}$)

$T_{\text{output}} = \{E_1 A_{i-1} D_0 D_1 B_i C_i V_i E_0 | V_i = 0, 2 \leq i \leq n\}$. 

Separation($T_{\text{output}}$, $\{C_0, C_1\}$, $T_{\text{tmp}}$)

$T_{\text{output}} = \{E_1 A_{i-1} D_0 D_1 B_i C_i V_i | V_i = 2 \leq i \leq n\}$. 

Copy($T_{\text{output}}$, $T_{\text{copy}}$)

$T_{\text{output}} = \{D_0(D)\setminus T, E_1 A_{i-1} D_0 D_1 B_i C_i V_i | V_i = 0, 2 \leq i \leq n\}$. 

Annealing($T_{\text{output}}$)

$T_{\text{output}} = \left\{ \begin{array}{l} E_1 A_{i-1} D_0 D_1 B_i C_i \\
D_0(D) \setminus T \\
\end{array} \right\} \forall i \leq n$. 

Cleavage($T_{\text{output}}$, $D_0(D)$)

$T_{\text{output}} = \left\{ \begin{array}{l} E_1 A_{i-1} D_0 D_1 B_i C_i \\
D_0(D) \setminus T \\
\end{array} \right\} \forall i \leq n$. 

Denaturation($T_{\text{output}}$)

$T_{\text{output}} = \{D_0, D_1, E_1 A_i D_0, D_1 B_i C_i | V_i = 0, 2 \leq i \leq n\}$. 

Step 5: Produce the output

\[ T_{\text{output}} = \{E_1A_{i-1}D_0|V_i = 0, 2 \leq i \leq n\}. \]
such that $F_k$ with DNA molecules $\bar{E}$.

Note that the operation $\text{ValueAssignment}$ introduced by Fujiwara et al. (2004) is used in Step 4. Although there is a bit difference for the DNA representations of binary numbers given by (1) and (4), the operation $\text{ValueAssignment}$ is still available for our setting. For readers’ convenience, we give a description of the operation $\text{ValueAssignment}$ as follows. Let

$$T_{\text{input}} = \{E_1A_1D_1D_1B_1C_1V_1E_0, E_1A_1D_1D_1B_1C_1V_1E_1| 1 \leq i \leq n-1\}.$$

where $V_i \in \{0, 1\}$. Then the operation $\text{ValueAssignment}_i(T_{\text{input}}, T_{\text{output}})$ produces

$$T_{\text{output}} = \{E_1A_1D_1D_1B_1C_1V_iE_0 | 1 \leq i \leq n\}.$$

where $V \in \{0, 1\}$ and all memory strands are set to the same value $V$. Let $T_C$ and $T_V$ be two auxiliary test tubes such that $T_C = \{T_0[C_1], T_V = \{C_1V_iE_0, V_0[C_1]\}$.

**Procedure** $\text{ValueAssignment}_i(T_{\text{input}}, T_{\text{output}})$

**Step 1:** Delete values from memory strands.

- **Copy**($T_C$, $T_V$)
- **Merge**($T_{\text{input}}, T_C$)
- $T_{\text{input}} = \{E_1A_1D_1D_1B_1C_1V_iE_0 | 1 \leq i \leq n\}$.
- **Annealing**($T_{\text{input}}$)
- $T_{\text{input}} = \left\{\left[\frac{E_1A_1D_1D_1B_1C_1V_iE_0}{C_1E_0}\right] | 1 \leq i \leq n\right\}$.
- **Cleavage**($T_{\text{input}}, C_1C_1$)
- $T_{\text{input}} = \left\{\left[\frac{E_1A_1D_1D_1B_1C_1V_iE_0}{C_0C_0}\right] | 1 \leq i \leq n\right\}$.
- **Denaturation**($T_{\text{input}}$)
- $T_{\text{input}} = \{C_0C_1, E_1A_1D_1D_1B_1C_1V_iE_0 | 1 \leq i \leq n\}$.
- **Separation**($T_{\text{input}}, C_1C_1, C_0C_0$)
- $T_{\text{input}} = \{E_1A_1D_1D_1B_1C_1 | 1 \leq i \leq n\}$.

**Step 2:** Assign values to memory strands.

- **Merge**($T_{\text{input}}, T_V$)
- $T_{\text{input}} = \{C_1V_iE_0, \frac{E_1A_1D_1D_1B_1C_1V_iE_0}{C_0C_0} | 1 \leq i \leq n\}$.
- **Annealing**($T_{\text{input}}$)
- $T_{\text{input}} = \left\{\left[\frac{E_1A_1D_1D_1B_1C_1V_iE_0}{C_0C_0}\right] | 1 \leq i \leq n\right\}$.
- **Denaturation**($T_{\text{input}}$)
- $T_{\text{input}} = \{E_1A_1D_1D_1B_1C_1V_iE_0 | 1 \leq i \leq n\}$.
- **Separation**($T_{\text{input}}, C_0C_0, T_C$)
- $T_{\text{input}} = \{E_1A_1D_1D_1B_1C_1V_iE_0 | 1 \leq i \leq n\}$.
- **Copy**($T_{\text{input}}, T_{\text{output}}$)
- $T_{\text{input}} = \{E_1A_1D_1D_1B_1C_1V_iE_0 | 1 \leq i \leq n\}$.

3. **Procedure for $F$ with DNA molecules**

Let $F$ be defined by (3). In this section we design a procedure with DNA molecules to check if there exists a $k \in \mathbb{N}$ such that $F^k(V^{(0)}) = (1, 1, \ldots, 1)$ for any given non-zero sequence $V^{(0)} = (V_1^{(0)}, V_2^{(0)}, \ldots, V_n^{(0)}) \in \{0, 1\}^n$. It is also
available for the general case $F^k(V^{(0)}) = \beta$ with $\beta \in \{0, 1\}^n$ after some minor modifications. Let

$$T_{\text{input}} = [S_i, i = 1, \ldots, n] = [E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0, i = 1, \ldots, n]$$

and

$$T_{\text{target}} = \{D\delta, E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 D\delta, i = 1, \ldots, n\}.$$ 

For $k \in \mathbb{N}$ let $V^{(i)} = F^i(V^{(0)}) = (V_1^{(i)}, V_2^{(i)}, \ldots, V_n^{(i)})$, the $i$th iteration of $F$ on $V^{(0)}$. The test tubes $T_{\text{value}}$ and $T_{\text{target}}$ are used to store the first and second bits of a binary number, respectively.

For $k = 1$ to $k = 2^n - 1$

Step 1: Evaluate $V^{(k)} = F(V^{(k-1)}) = F(V^{(0)})$. Note that $V^{(k)} = F(V^{(k-1)}) = F_1 \circ F_2(V^{(k-1)})$, $F_2$ is evaluated by the first sub-step [1-1]-[1-9]. $F_1$ is then evaluated by performing the operation of CycleShift in the sub-step [1-1].

[1-1] Separation($T_{\text{input}}, (B_1), T_{\text{target}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid 2 \leq i \leq n\}. \]

[1-2] Separation($T_{\text{input}}, (B_1), T_{\text{target}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid 3 \leq i \leq n\}. \]

[1-3] Copy($T_{\text{input}}, T_{\text{target}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid i \neq 2, 1 \leq i \leq n\}. \]

[1-4] Merge($T_{\text{input}}, T_{\text{temp}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid i = 1, 2\}. \]

[1-5] Copy($T_{\text{target}}, T_{\text{temp}}$)  \[ \Rightarrow T_{\text{temp}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid i \neq 2, 1 \leq i \leq n\}. \]

[1-6] Merge($T_{\text{temp}}, T_{\text{temp}}$)  \[ \Rightarrow T_{\text{temp}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid i = 1, 2\}. \]

[1-7] Separation($T_{\text{input}}, C_1 E_0, T_{\text{temp}}$)  \[ \Rightarrow \text{if Detect}(T_{\text{input}}) \text{ is } "\text{no}" \text{, then ValueAssignment}(T_{\text{input}}, T_1); \text{ else if Detect}(T_{\text{temp}}) \text{ is } "\text{no}" \text{, then ValueAssignment}(T_{\text{target}}, T_1) \text{ else ValueAssignment}(T_{\text{target}}, T_1) \]

[1-9] Merge($T_{\text{input}}, T_1$)  \[ \Rightarrow T_1 = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \}. \]

[1-10] Discard($T_{\text{target}}, T_1$, and ($T_{\text{temp}}$)  \[ \Rightarrow T_{\text{temp}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid i \neq 2, 1 \leq i \leq n\}. \]

[1-11] CycleShift($T_{\text{input}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0, 3 \leq i \leq n\}. \]

[1-12] Discard($T_{\text{input}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0, 3 \leq i \leq n\}. \]

[1-13] Merge($T_{\text{input}}, T_{\text{temp}}$)  \[ \Rightarrow T_{\text{temp}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0, 3 \leq i \leq n\}; \]

Step 2: Check whether or not $V^{(k)} = F^k(V^{(0)}) = (1, 1, \ldots, 1)$.

[2-1] Separation($T_{\text{input}}, (C_1 E_0), T_{\text{temp}}$)  \[ \Rightarrow T_{\text{temp}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid V^{(k)} = 0, 1 \leq i \leq n\}. \]

[2-2] if Detect($T_{\text{temp}}$) is “no”, then end the procedure; else continue the following operations.

[2-3] Merge($T_{\text{input}}, T_{\text{temp}}$)  \[ \Rightarrow T_{\text{input}} = \{E_i A_i D_i B_i C_i \gamma_i^{(0)} E_0 \mid V^{(k)} \neq 1 \}. \]
All implementations described in (Fujiwara et al., 2004) remain effective with this improved DNA representation.

by Fujiwara et al. (2004) for a binary number so that it is more able to deal with some mathematical problems.

In this paper, we improve the DNA representation given (e.g. Adleman, 1994; Lipton, 1995; Ouyang et al., 1997) and primitive operations, such as logic or arithmetic operations problems. Since then, lots of papers on DNA computing have been published. They are mainly involved in NP problems.

4. Conclusions

In 1994, Adleman (1994) published a pioneering work to show that DNA can be used to perform mathematical problems. This has led several researchers (e.g. Adleman, 1994, 1995; Lipton, 1995; Ouyang et al., 1997) to conclude that the complexity aspects of DNA algorithms will limit their applicability. The other is the error rate in the biological
manipulations (e.g. Li et al., 2003). However, we believe that these problems will be overcome as biological technique will be continually developed in the future.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments and suggestions that lead to the improvement of the manuscript. This project is supported by Bio-X DNA Computer Consortium No. 03DZ14025. The second author is also supported by the National Science Foundation of China #10371043 and Shanghai Priority Academic Discipline.

References