



System identification: DNA computing approach

Ching-Huei Huang^a, Horn-Yong Jan^b, Chun-Liang Lin^{a,*}, Chia-Soon Lee^a

^a Department of Electrical Engineering, National Chung Hsing University, 250, Kuo Kuang Road, Taichung, 402, Taiwan, ROC

^b Graduate Institute of Electrical and Communications Engineering, Feng Chia University, 100 Wenhwa Road, Taichung, 40724, Taiwan, ROC

ARTICLE INFO

Article history:

Received 20 June 2008

Received in revised form

15 December 2008

Accepted 26 January 2009

Available online 27 February 2009

Keywords:

DNA computing algorithm (DNACA)

Electron–ion interaction potential (EIIP)

Systems identification

ABSTRACT

A DNA computing algorithm (DNACA) with an electron–ion interaction potential (EIIP) decoding scheme is proposed to identify a class of transfer functions. The DNACA includes enzyme and virus operators which provide a highly modular, flexible, and accurate self-organizing structure environment. Simulation study based on De Jong's test functions show its superior performance when compared with the improved and standard genetic algorithms (GAs).

© 2009 ISA. Published by Elsevier Ltd. All rights reserved.

1. Introduction

Recently developed DNA computing algorithms (DNACAs) have inspired new methods that can simultaneously solve the parameter and structure optimization problems. The DNACAs based on the concept of bimolecular evolution was first developed by Adleman [1]. Maley [2] further detailed this kind of algorithm in terms of chemical processes and computer programming.

The operational features inherent in DNACAs make the algorithms in DNA computers which are over a billion times possible to implement more computationally efficient than the conventional computers. The massively parallel nature of DNA in those computers means that computing may be millions or billions of times beyond today's supercomputers (Forbes [3]).

In this paper, DNACA with an electron–ion interaction potential (EIIP) decoding scheme is proposed to identify a class of transfer functions (Cotic et al. [4]). Accuracy of the proposed approach with or without frameshift operators is verified first by numerical testing of De Jong's [5] test functions. With regard to system identification, the DNA strings are created to represent the transfer model in which codon is used to represent coefficients of the denominator and numerator polynomials. Using frameshift operators (enzyme and virus), the order and numeric values of the transfer model can be refined to fit the identified object. Verification of the simulated results show that the proposed method is quite well-performed, even for the systems with wide dynamic responses.

Applying DNA computing techniques carries distinct advantages and has great potential, as outlined here:

- Current molecular biotechnology is available for implementing DNA computing operators (Chen et al. [6], and Jan et al. [7]).
- Compared to conventional computers, certain operations can make DNA computing over a billion times more energy efficient. The massively parallel nature of DNA means that computing its sequence may be millions or billions of times beyond the reach of today's supercomputers [3].
- More capacity for data storage is available using DNA sequences. A gram of DNA, for example contains approximately 2.1×10^{21} bases, which has an information content of approximately 4.2×10^{21} bits, far more than the 200 petabyte storage of all the digital magnetic tape produced (Tsuboi et al. [8]).

This paper describes how a model selection problem can be solved by a synthesis of qualitative and quantitative techniques using a model structure and numeric computations. Accuracy can be verified by numerical analyses of De Jong's test functions through the proposed DNACA with or without frameshift operators. For system identification, an imitating model is constructed from DNA strings in which codons were used to represent the coefficients of the denominator and numerator polynomials. Using special frameshift operators, the order and the numeric values of the parameters can refine the properties of the modeling system more precisely.

2. Biological computation algorithms

A DNA strand consists of two complimentary strings in which four nucleotide base: adenine (A), cytosine (C), guanine (G), and thymine (T) are arranged in various combinations. Nucleotides are

* Corresponding author. Tel.: +886 4 22851549x708.

E-mail address: chunlin@dragon.nchu.edu.tw (C.-L. Lin).

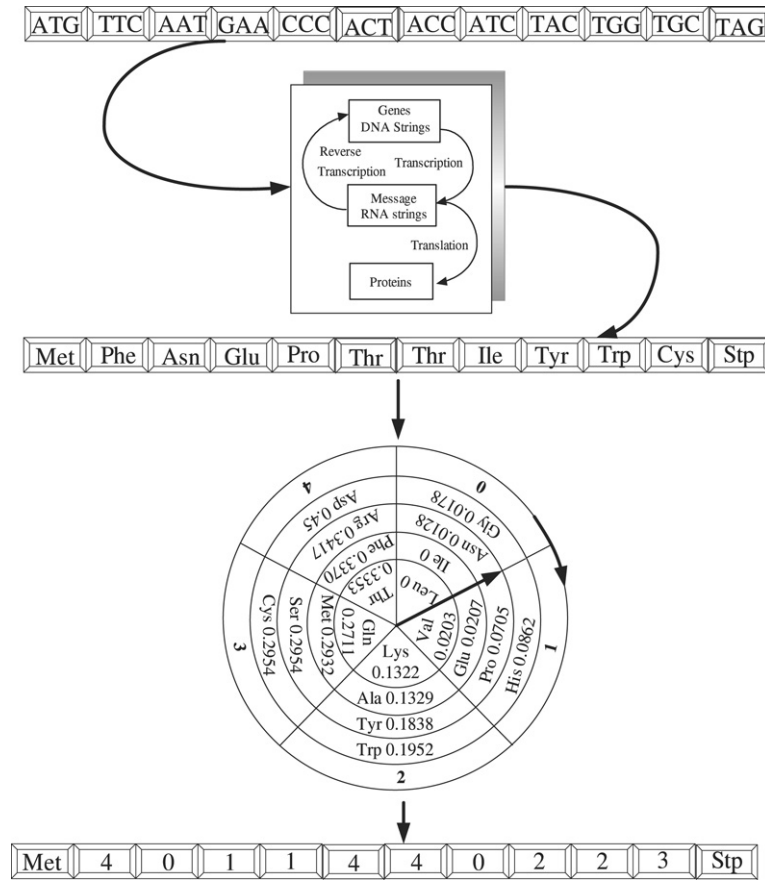


Fig. 1. Normalized EIIP wheel of amino acids.

paired along with two strings. For the process of computation, one DNA string is separated from the other through chemical process. As the other string is a perfect compliment, only one string is needed to further operations.

DNACAs consist of several operators in which a crossover is used to generate a new strand that will retain beneficial features from the parent generation. This exploits current genetic potential. If the population does not contain enough encoded information to solve a particular problem, none of the mixing strands can produce a satisfactory solution. For this reason, the mutation operator capable of spontaneously generating new strands provides a mechanism to maintain the population’s diversity (Goldberg [9]).

In addition to the mutation operator, two frameshift operators are depicted in what follows.

2.1. Enzyme mutation

The enzyme mutation works to separate and connect two DNA substrings while removing a section of codon. The newly formed codon breaks away from the doped position and the reacted remainder of codon are joined together to form a new one. The direct consequence of this operation is that the resulting DNA string loses some information in the next generation.

2.2. Virus mutation

From the biological viewpoint, the major cause of disease is usually that a virus has intruded into the human body, modifying the original codon of DNA strings and reproducing the infected strings. Through this mechanism, the inserted DNA string makes an incursion into its parent’s string to form a new one. The direct consequence of this operation is that the resulting DNA string gains extra information in the next generation.

3. Transfer function modeling based on DNACA

3.1. Configurable transfer functions

Consider a class of transfer function modeled as follows:

$$\widehat{C}_E(s) = \widehat{C}_C \frac{\prod_{i=1}^{n_n} (s + \widehat{C}_{N1,i}) \prod_{i=1}^{m_n} (s^2 + \widehat{C}_{N2,i}s + \widehat{C}_{N3,i})}{\prod_{i=1}^{n_d} (s + \widehat{C}_{D1,i}) \prod_{i=1}^{m_d} (s^2 + \widehat{C}_{D2,i}s + \widehat{C}_{D3,i})} \quad (1)$$

Let $\tilde{n}_n = n_n + 2m_n$ and $\tilde{n}_d = n_d + 2m_d$ with $\tilde{n}_d \geq \tilde{n}_n$. $\widehat{C}_E(s)$ is said to be proper if $\tilde{n}_d \geq \tilde{n}_n$ and is improper if $\tilde{n}_d < \tilde{n}_n$.

In traditional system identification methodologies, the least mean square error scheme and some iterations are commonly adopted to build the transfer function model so that it ultimately mimics the relationship. However, the most desirable transfer function is not usually the one with minimal order. In the following sections, the DNACA is used to reconstruct the model and module of the estimated system.

3.2. DNA coding scheme

Biologists have discovered 20 amino acids encoded by 64 codons, which are expressed in the versatile combinations of {A, C, T, G}. As in biological DNA, a gene starts with the codon ATG and ends with the codon TAA, TAG or TGA.

Electron–ion interaction potential (EIIP) is applied here instead of binary coding system. In the presented method, each amino acid is represented by a specific number referred to as the unique electron–ion interaction potential and it is irrespective of its

Table 1
Translation of DNA codons.

Amino acid	Three-letter code	One-letter code	Code of amino acid	Electron–ion interaction potential values	
Alanine	ALA	Ala(12)	A	GCT GCC GCA GCG	0.0373
Arginine	ARG	Arg(11)	R	CGT CGC CGA CCG AGA AGG	0.0959
Aspartic	ASP	Asp(17)	D	GAT GAC	0.1263
Asparagine	ASN	Asn(15)	N	AAT AAC	0.0036
Cysteine	CYS	Cys(19)	C	TGT TGC	0.0829
Glutamic	GLU	Glu(18)	E	GAA GAG	0.0058
Glutamine	GLN	Gln(14)	Q	CAA CAG	0.0761
Glycine	GLY	Gly(13)	G	GGT GGC GGA GGG	0.0050
Histidine	HIS	His(5)	H	CAT CAC	0.0242
Isoleucine	ILE	Ile(2)	I	ATT ATC ATA	0.0000
Leucine	LEU	Leu(9)	L	TTA TTG CTT CTC CTA CTA	0.0000
Lysine	LYS	Lys(16)	K	AAA AAG	0.0371
Methionine	MET	Met(3)	M	ATG	0.0823
Phenylalanine	PHE	Phe(1)	F	TTT TTC	0.0946
Proline	PRO	Pro(7)	P	CCT CCC CCA CCG	0.0198
Serine	SER	Ser(10)	S	TCT TCC TCA TCG AGT AGC	0.0829
Threonine	THR	Thr(8)	T	ACT ACC ACA ACG	0.0941
Tryptophan	TRP	Trp(20)	W	TGG	0.0548
Tyrosine	TYR	Tyr(4)	Y	TAT TAC	0.0516
Valine	VAL	Val(6)	V	GTT GTC GTA GTG	0.0057

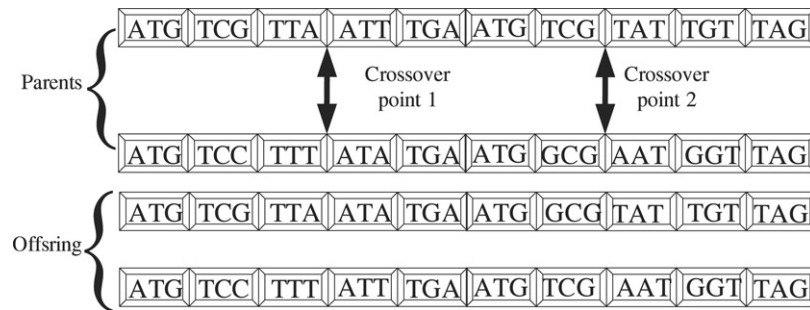


Fig. 2. Multi-point crossover operation.

position in a DNA string. Moreover, these numbers are essential to build a physical and mathematical model which interprets protein sequences linear information by using signal analysis methods.

The EIIP values for 20 amino acids and four nucleotides are summarized in Table 1. Normalization is then used to build an orthogonal based on these values, as shown in Fig. 1. It allows all amino acids to be divided into five groups based on their EIIP.

Each DNA string is encoded here as a vector in the following form:

$$\vec{S} = [\vec{S}_1 \quad \vec{S}_2 \quad \dots \quad \vec{S}_\mu]. \quad (2)$$

The individuals are defined as in Box I, where \vec{S}_{IND} consists of the structure information \vec{S}_{INF} and the parameter information \vec{S}_{PARA} . \vec{S}_{INF} represents the coding information for the system gain, denominator polynomial and numerator polynomial based on the combination of different numbers of amino acids. To fully specify the required information for characterizing the denominator or numerator polynomial, excluding lead and end codons, each needs 6 codon triplets denoted by $\vec{S}_{\text{INF},G,N,D(A)} \sim \vec{S}_{\text{INF},G,N,D(F)}$ which constitute the elements for the translation process. The sequence \vec{S}_{PARA} includes a constant term $S_{\text{PARA},G}$, a numerator polynomial $\vec{S}_{\text{PARA},N}$ and a denominator polynomial $\vec{S}_{\text{PARA},D}$, based on various combinations of amino acids. S_{STA} and S_{STP,S_s} with $S_s \triangleq \text{INF}, D, N$ were used to identify lead and end codons. Subscript notes (such as “f” or “s”) are defined as the polynomials with the first- or second-order polynomial.

Normalization is performed based on the categorization of EIIP; the values for 20 amino acids and four nucleotides are summarized in Table 1. Accordingly, the decoded parameters are calculated,

excluding lead and end codon, as follows

$$r \left(\frac{\sum_{i=1}^l \text{DNA}_{AA, EIIP}^{(i)} \times 5^{i-1}}{N} \right) - M \quad (3)$$

where the constants r , M and N control the range and resolution of the parameters; $\text{DNA}_{AA, EIIP}^{(i)}$ is the number of the i th amino acid of \vec{S}_{PARA} ; l is the length of the corresponding parameters.

3.3. Crossover operation

It causes an exchange of DNA information between two strings which have the same length via random decision in the mating pool and provide a mechanism to produce and match desirable qualities. Several string couples in the mating pool are randomly selected according to the crossover rate G_c and the multi-point scheme, as shown in Fig. 2.

3.3.1. Crossover information fragment

See Box II.

3.3.2. Crossover parameter fragment

See Box III, where L is length of the corresponding fragment, $\rho_{1,2}$ indicate the position of the crossover points. $S_{\text{PARA},i}$ and $i = N_f, D_s, N_f$ or D_s are defined as in Box I. DNA_{RES} specifies the resolution for the corresponding term where each unit (codon) complies 3 nucleotides and τ is defined to specify the number of DNA_{RES} for the parameter resolution.

$$\begin{aligned}
 \vec{S}_{IND} &= [\vec{S}_{INF} \ \vec{S}_{PARA}], \quad IND = 1, \dots, \mu, \\
 S_{STA} &\triangleq ATG, S_{STP} \triangleq \{TAG, TGA, TAA\}, \\
 \vec{S}_{INF} &= [\vec{S}_{INF_G} \ \vec{S}_{INF_D} \ \vec{S}_{INF_N}], \\
 \vec{S}_{INF_G} &= [S_{STA} \ S_{INF_G(1)} \ S_{STP} \ S_{STP_INF}], \\
 \vec{S}_{INF_D} &= [S_{STA} \ S_{INF_D(1)} \ \dots \ S_{INF_D(7)} \ S_{STP} \ S_{STP_INF}], \\
 \vec{S}_{INF_N} &= [S_{STA} \ S_{INF_N(1)} \ \dots \ S_{INF_N(7)} \ S_{STP} \ S_{STP_INF}], \\
 S_{STP_INF} &\triangleq TGATGA, \\
 \vec{S}_{PARA} &= [S_{PARA_G} \ \vec{S}_{PARA_D} \ \vec{S}_{PARA_N}], \\
 S_{PARA_G} &= [S_{STA} S_{\hat{C}} S_{STP} S_{STP_G}], \\
 \vec{S}_{PARA_D} &= [\vec{S}_{PARA_D_f} \ \vec{S}_{PARA_D_s}] = [S_{STA} S_{\hat{C}_{N1,1}} S_{STP} S_{STA} S_{\hat{C}_{N1,2}} \dots S_{STA} S_{\hat{C}_{N1,m}} S_{STP} S_{STA} S_{\hat{C}_{N2,1}} S_{STP} S_{STA} S_{\hat{C}_{N3,1}} \dots S_{\hat{C}_{N3,m}} S_{STP} S_{STP_N}], \\
 \vec{S}_{PARA_N} &= [\vec{S}_{PARA_N_f} \ \vec{S}_{PARA_N_s}] = [S_{STA} S_{\hat{C}_{D1,1}} S_{STP} S_{STA} S_{\hat{C}_{D1,2}} \dots S_{STA} S_{\hat{C}_{D1,d}} S_{STP} S_{STA} S_{\hat{C}_{D2,1}} S_{STP} S_{STA} S_{\hat{C}_{D3,1}} \dots S_{\hat{C}_{D3,d}} S_{STP} S_{STP_D}], \\
 S_{STP_G} &\triangleq TGA, S_{STP_D} \triangleq TAA, S_{STP_N} \triangleq TAG
 \end{aligned}$$

Box I.

$$\begin{aligned}
 \left\{ \begin{array}{l} \vec{S}_{c(IND)}^t \\ \vec{S}_{c(IND+1)}^t \end{array} \right\} &= O_{\{G_c\}} \left(\left\{ \begin{array}{l} \vec{S}_{(IND)}^t \\ \vec{S}_{(IND+1)}^t \end{array} \right\} \right), \quad \forall IND \in \{1, 2, \dots, \mu - 1\} \\
 \Rightarrow \left\{ \begin{array}{l} [S_{INF_i(IND+1),2}, S_{INF_i(IND+1),3}, \dots, S_{INF_i(IND+1),\rho_1}, S_{INF_i(IND),(\rho_1+1)}, \\ \dots, S_{INF_i(IND),\rho_2}, S_{INF_i(IND+1),(\rho_2+1)}, \dots, S_{INF_i(IND+1),L}] \\ [S_{INF_i(IND),2}, S_{INF_i(IND),3}, \dots, S_{INF_i(IND),\rho_1}, S_{INF_i(IND+1),(\rho_1+1)}, \\ \dots, S_{INF_i(IND+1),\rho_2}, S_{INF_i(IND),(\rho_2+1)}, \dots, S_{INF_i(IND),L}] \end{array} \right\}, \quad 1 \leq \rho_1 < \rho_2 \leq L, \\
 \left\{ \begin{array}{l} L = DNA_{RES}, \quad \text{if } i = G \\ L = 7 \cdot DNA_{RES}, \quad \text{if } i = N \text{ or } D \end{array} \right.
 \end{aligned}$$

Box II.

$$\begin{aligned}
 \Rightarrow \left\{ \begin{array}{l} [S_{PARA_i(IND+1),1}, S_{PARA_i(IND+1),2}, \dots, S_{PARA_i(IND+1),\rho_1}, S_{PARA_i(IND),(\rho_1+1)}, \\ \dots, S_{PARA_i(IND),\rho_2}, S_{PARA_i(IND+1),(\rho_2+1)}, \dots, S_{PARA_i(IND+1),L}] \\ [S_{PARA_i(IND),1}, S_{PARA_i(IND),2}, \dots, S_{PARA_i(IND),\rho_1}, S_{PARA_i(IND+1),(\rho_1+1)}, \\ \dots, S_{PARA_i(IND+1),\rho_2}, S_{PARA_i(IND),(\rho_2+1)}, \dots, S_{PARA_i(IND),L}] \end{array} \right\}, \quad 1 \leq \rho_1 < \rho_2 \leq L, \\
 \left\{ \begin{array}{l} L = \tau DNA_{RES}, \quad \text{if } i = G \\ L = \tau DNA_{RES} S_{INF_j,4} \quad \text{and } j = N \text{ or } D, \quad \text{if } i = N_f \text{ or } D_f \\ L = \tau DNA_{RES} S_{INF_j,5} \quad \text{and } j = N \text{ or } D, \quad \text{if } i = N_s \text{ or } D_s \end{array} \right.
 \end{aligned}$$

Box III.

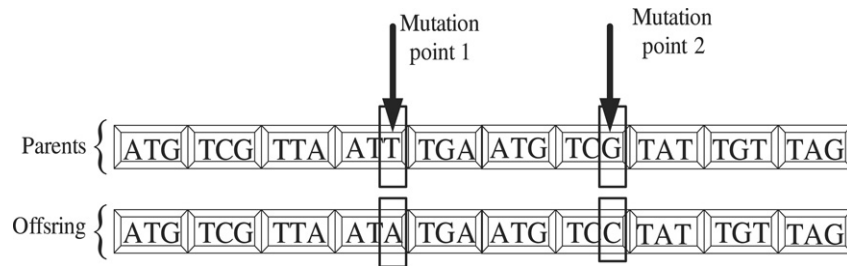


Fig. 3. Multi-point mutation operation.

3.4. Mutation operation

A multi-point mutation is applied according to the mutation rate G_m and the principle of codon exchange $A \leftrightarrow T$ and $G \leftrightarrow C$ to ensure that all points in the search space could be ultimately reached. A mutation would simultaneously cause data alternation on the two fragments, in Fig. 3.

3.4.1. Mutation information fragment

See Box IV.

3.4.2. Mutation parameter fragment

See Box V,

3.5. Enzyme and virus mutation

Enzyme and virus operators are randomly applied to alter the structure and parameters of the DNA strings during translation and evolution. Through the end codon, the segment corresponding to the first- and second-order polynomials that being selected for insertion or deletion of codon can be identified. Enzyme and virus operators also apply to the parameter fragment to generate a variant resolution of the parameters. The two operators work by

$$\vec{S}_{m(IND)}^t = O_{\{G_m\}}(\vec{S}_{c(IND)}^t), \quad \forall IND \in \{1, \dots, \mu - 1\}$$

$$\vec{S}_{m(INF_i(IND),k)}^t = \begin{cases} \vec{S}_{c(INF_i(IND),k)}^t, & k \in \{1, \dots, p - 1, p + 1, \dots, L\} \\ \vec{S}_{c(INF_i(IND),k)}^t, & k = p, \end{cases} \quad 1 \leq p < L.$$

Box IV.

$$\vec{S}_{m(PARA_i(IND),k)}^t = \begin{cases} \vec{S}_{c(PARA_i(IND),k)}^t, & k \in \{1, \dots, p - 1, p + 1, \dots, L\} \\ \vec{S}_{c(PARA_i(IND),k)}^t, & k = p, \end{cases} \quad 1 \leq p < L$$

where p indicates the location of the mutation point.

Box V.

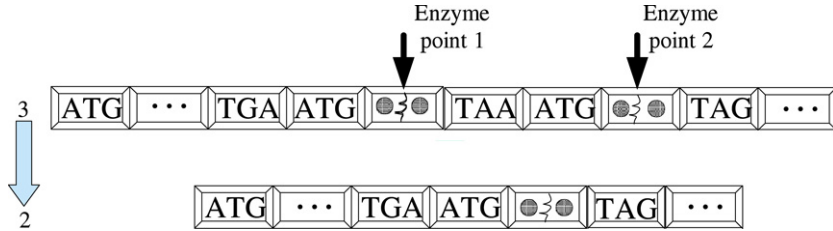


Fig. 4. Enzyme mutation operation.

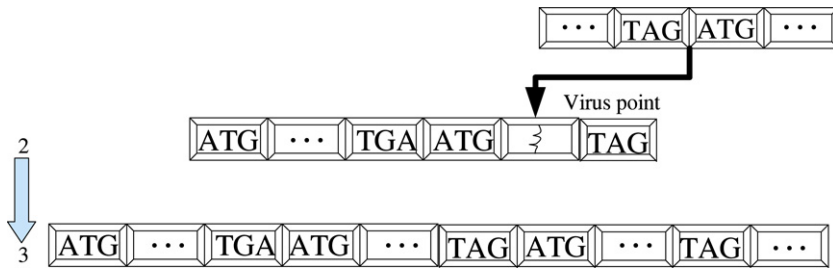


Fig. 5. Virus mutation operation.

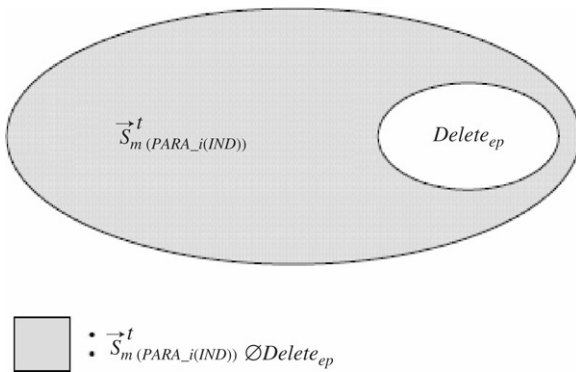


Fig. 6a. Illustrations of enzyme mutation operations.

randomly altering n consecutive string units with each string's unit representing a control coefficient based on the deletion rate G_e and the insertion rate G_v , respectively, as shown in Figs. 4 and 5.

3.5.1. Enzyme mutation

The enzyme operation works as a bit deletion mutation, it can be divided into separation and connection. In the former, the reacting molecules go through chemical changes rapidly. The newly formed molecules break away from the doped position, as shown in Fig. 6a.

$$\{\vec{S}_{e(PARA_i(IND))}^t\} \triangleq O_{\{G_e\}}\{\vec{S}_{m(PARA_i(IND))}^t\}, \quad \forall IND \in \{1, 2, \dots, \mu_e\}$$

$$\triangleq \{\vec{S}_{m(PARA_i(IND))}^t \setminus Delete_{ep}\},$$

where

$$\{S_{m(PARA_i(IND),2)}^t, S_{m(PARA_i(IND),3)}^t, \dots, S_{m(PARA_i(IND),L-3)}^t\}, \quad (4)$$

$$Delete_{ep} = \vec{S}_{m(PARA_i(IND),\rho_j^e)}^t, \quad 1 < \rho_j^e < L,$$

$$\vec{S}_{m(PARA_i(IND),\rho_j^e)}^t \in X_k$$

and $X_k \in \vec{S}_{m(PARA_i(IND))}^t$, $X_k = (x_1x_2x_3)$, $x_i \in \{A, G, T, C\}$
 \emptyset means the excision of enzyme mutation operations.

where the length L is defined as in Box I.

3.5.2. Virus mutation

From a biological viewpoint, one of the major causes of disease is that a virus intrudes into the human body, it works as a bit insertion mutation, as shown in Fig. 6b.

$$\{\vec{S}_{v(PARA_i(IND))}^t\} \triangleq O_{\{G_v\}}\{\vec{S}_{m(PARA_i(IND))}^t\}, \quad \forall IND \in \{1, 2, 3, \dots, \mu_v\}$$

$$\triangleq \{\vec{S}_{m(PARA_i(IND))}^t \oplus Insert_{vp}\}$$

where

$$\{S_{m(PARA_i(IND),2)}^t, \dots, Insert_{vp}, \dots, S_{m(PARA_i(IND),L+3)}^t\}, \quad (5)$$

$$Insert_{vp} = \vec{S}_{m(PARA_i(IND),\rho_i^v)}^t, \quad 1 < \rho_i^v < L,$$

$$\vec{S}_{m(PARA_i(IND),\rho_i^v)}^t \in X_k, X_k = (x_1x_2x_3), x_i \in \{A, G, T, C\}$$

\oplus means the combination of the virus mutation operations

where the length L is defined as in Box I.

Table 2

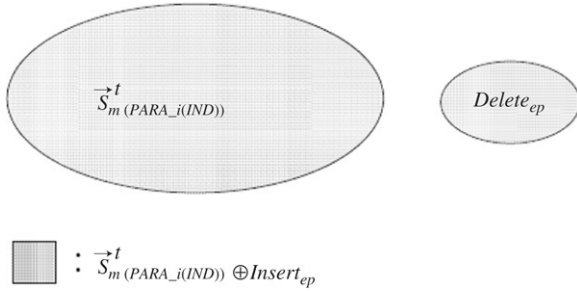
Results searched by the proposed DNACA, the standard GA and the improved GA for De Jong's function tests.

De Jong's test functions (converging generation)	Proposed DNACA	Standard GA	Improved GA
$D_1(x)$	1.0000(25)	1.0000(225)	1.0000(100)
$D_2(x)$	1.0000(20)	0.6393(>450)	0.9707(>045)
$D_3(x)$	0.9995(40)	0.8037(>450)	0.8349(>450)
$D_4(x)$	1.0000(25)	1.0000(200)	1.0000(>450)
$D_5(x)$	1.0000(45)	0.7297(>500)	1.0000(75)

Table 3

Comparison of DNACA efficiency with/without frameshift operators.

Test functions	D_4 with two parameters		D_5 with three parameters	
	Condition			
	With frameshift operators	Without frameshift operators	With frameshift operators	Without frameshift operators
Fitness value	1.000	0.991	1.000	0.6465
Converging generation	20	50	45	45
Length	8, 3 codons	6, 6 codons	1, 2, 2 codons	6, 6, 6 codons
DNA string and decimal values of parameters	CATTTCTCAGTT	ACGGCGCCG	TTA(0)	TTTTTAGACCT
	AAATATGAAGTC	GCCCACGAA	TTGGGA(0)	TATAAG(−0.0413)
	(−31.4014)	(−31.5653)	TACTGG(0)	ACCCAGAACTAT
	GCACAGTCT	AGGGCCAAA		TGGTGG(−0.0282)
	(31.6239)	GCGCAGATG		TATTGGGAAGCT
	(31.3896)		GCAGCA(−0.0164)	

**Fig. 6b.** Illustrations of virus mutation operations.

Succeeding to one of the frameshift mutations there is a translation process to recount the length of the parameter fragment and examine the proper condition for the information fragment that ensures not only a realizable controller but maintains consistency of the two fragments. After each operation, a translation process is applied so that the corresponding parameter fragment is updated accordingly.

3.6. Objective function

The objective function for the current problem is defined as follows

$$J = \sum_i w_i J_i \quad (6)$$

where w_i is the weighting factor for the corresponding term, three objective functions are chosen:

$$J_1(\vec{S}_{IND}) : G_e^{low} = \left| \frac{Mag(\hat{P}_E(j\omega_{lf}))}{10^{-\exp(\log(\omega_{lf}))}} - \frac{Mag(P_0(j\omega_{lf}))}{10^{-\exp(\log(\omega_{lf}))}} \right| \quad (7)$$

$$J_2(\vec{S}_{IND}) : M_e = \sum_{i=1}^{f_t} |\ln(Mag(\hat{P}_E(j\omega_i))) - \ln(Mag(P_0(j\omega_i)))| \quad (8)$$

$$J_3(\vec{S}_{IND}) : PA_e = \sum_{i=1}^{f_t} |phase(\hat{P}_E(j\omega_i)) - phase(P_0(j\omega_i))| \quad (9)$$

and ω_{lf} represents the lowest frequency; $Mag(\cdot)$ and $phase(\cdot)$ denote, respectively, the magnitude and phase of the frequency

response obtained at ω_i ; f_t is the total sampling number over the testing frequency interval. Here, J_1 denotes the magnitude of gain error at the lowest frequency; whereas J_2 and J_3 calculate, respectively, the overall errors of the magnitude and phase over the entire spectrum.

Although a complicated transfer function model generally enables superior performance, a simpler structure with at least acceptable performance is more meaningful from the practical viewpoint. Therefore, the objective function considered should contain not only system performance indices but also structure information. The general form of the performance index is defined as follows

$$F' = (1 - \beta)J_{ind} + \beta g_{ind} \quad (10)$$

where

$$J_{ind}(\vec{S}_{IND}) = \sum_{i=1}^3 \alpha_i J_i(\vec{S}_{IND}) \quad (11)$$

$$g_{ind}(\vec{S}_{IND}) = \frac{\tilde{n}_d^{IND}}{\max_{1 \leq IND \leq \mu} \tilde{n}_d^{IND}} \quad (12)$$

and F' is made up of the individual index J_i with α_i representing the corresponding weighting factor and β representing the desired emphasis on control complexity; \tilde{n}_d^{IND} is the orders of the INDth's denominator and it can be derived by counting the numbers of the leading and ending codon except for the first leading and final ending codon. The flowchart of DNACA for the optimal solution searching is illustrated in Fig. 7.

4. Cases study

Case 1: For De Jong's functions testing

To examine applicability and efficiency of the proposed DNACA, the following De Jong's test functions (De Jong's [5], Hanaki et al. [10], Frank et al. [11]) are considered:

$$D_1 = \sum_{i=1}^3 x_i^2, \quad -5.12 \leq x_i \leq 5.12 \quad (13)$$

where D_1 is a simple 3D parabola with a spherical constant-cost contour.

$$D_2 = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2, \quad -2.048 \leq x_i \leq 2.048 \quad (14)$$

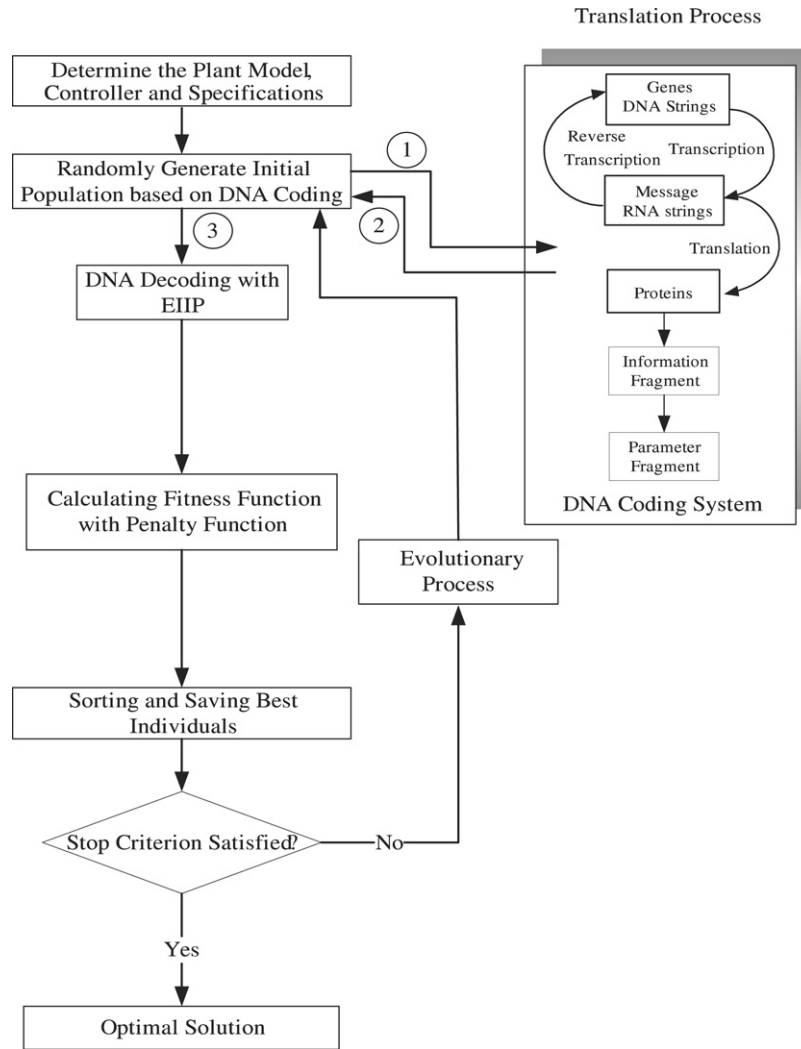


Fig. 7. Flowchart of DNACA algorithm for the optimal solution searching.

where D_2 is a typical test function to deal with the optimization. The point (1,1) is a minimum of zero—it is difficult to perceive because a deep parabolic valley is along with the relation $x_2 = x_1^2$.

$$D_3 = \sum_{i=1}^3 ix_i^4 + \text{Gauss}(0, 1), \quad -1.28 \leq x_i \leq 1.28 \quad (15)$$

where D_3 is a continuous, convex, unimodal, and 3D quadric which function with zero-mean Gaussian noise.

$$D_4 = 0.002 - \sum_{i=1}^{25} \frac{1}{j + \sum_{i=1}^2 (x_i - a_{ij})^6}, \quad -65.356 \leq x_i \leq 65.356 \quad (16)$$

where D_4 is a multi-modal function; it has 25 local minimums lying approximately at the points a_{ij} , $i = 1, 2$ and $j = 1, \dots, 25$.

$$D_5 = \sum_{i=1}^3 [x_i^2 - 10 \cos(2\pi x_i) + 10], \quad -5.12 \leq x_i \leq 5.12 \quad (17)$$

where D_5 is a generalized Rastrigin's function. It has multiple local optimum that can be used to examine the efficiency of the optimization algorithms.

The corresponding fitness functions F_D^1 for $D_{1,2,3,5}$ and F_D^2 for D_4 are given, respectively, as

$$F_D^1 = \frac{1}{1 + D_i(x)}, \quad i = 1, 2, 3, 5; \quad F_D^2 = \frac{1}{(|1 - D_4| + 1)}. \quad (18)$$

For each test function, the iteration and population numbers are set to be 50 and 10 respectively. The probability rates of crossover and mutation are 0.8 and 0.2, respectively. Both of the probability rates of enzyme and virus are 0.4. The results of the solution are searched by using the proposed DNACA as summarized in Table 2. Taking D_4 as the example, the efficiency of DNACA with/without frameshift operators is illustrated as in Table 3. Convergence of the fitness values are shown in Figs. 8 and 9 is displayed the values of D_4 and D_5 . It is seen that the DNACA can produce not only the accurate solutions, it simultaneously prevents the redundant parts of candidate strings from being repeated in the evolutionary process. These results demonstrate superior efficiency and performance of the present algorithm while comparing to other approaches.

We proceed to show how the DNACA can be used to identify the following system model:

$$P_0 = \frac{1.038 \left(\frac{s}{300} + 1 \right) \left(\frac{s^2}{0.0911^2} + \frac{2 \times 1.17s}{0.0911} + 1 \right) \left(\frac{s^2}{0.7587^2} + \frac{2 \times 1.129s}{0.7587} + 1 \right)}{s \left(\frac{s}{4.012} + 1 \right) \left(\frac{s^2}{0.1068^2} + \frac{2 \times 2.712s}{0.1068} + 1 \right) \left(\frac{s^2}{2.168^2} + \frac{2 \times 1.59s}{2.168} + 1 \right)}. \quad (19)$$

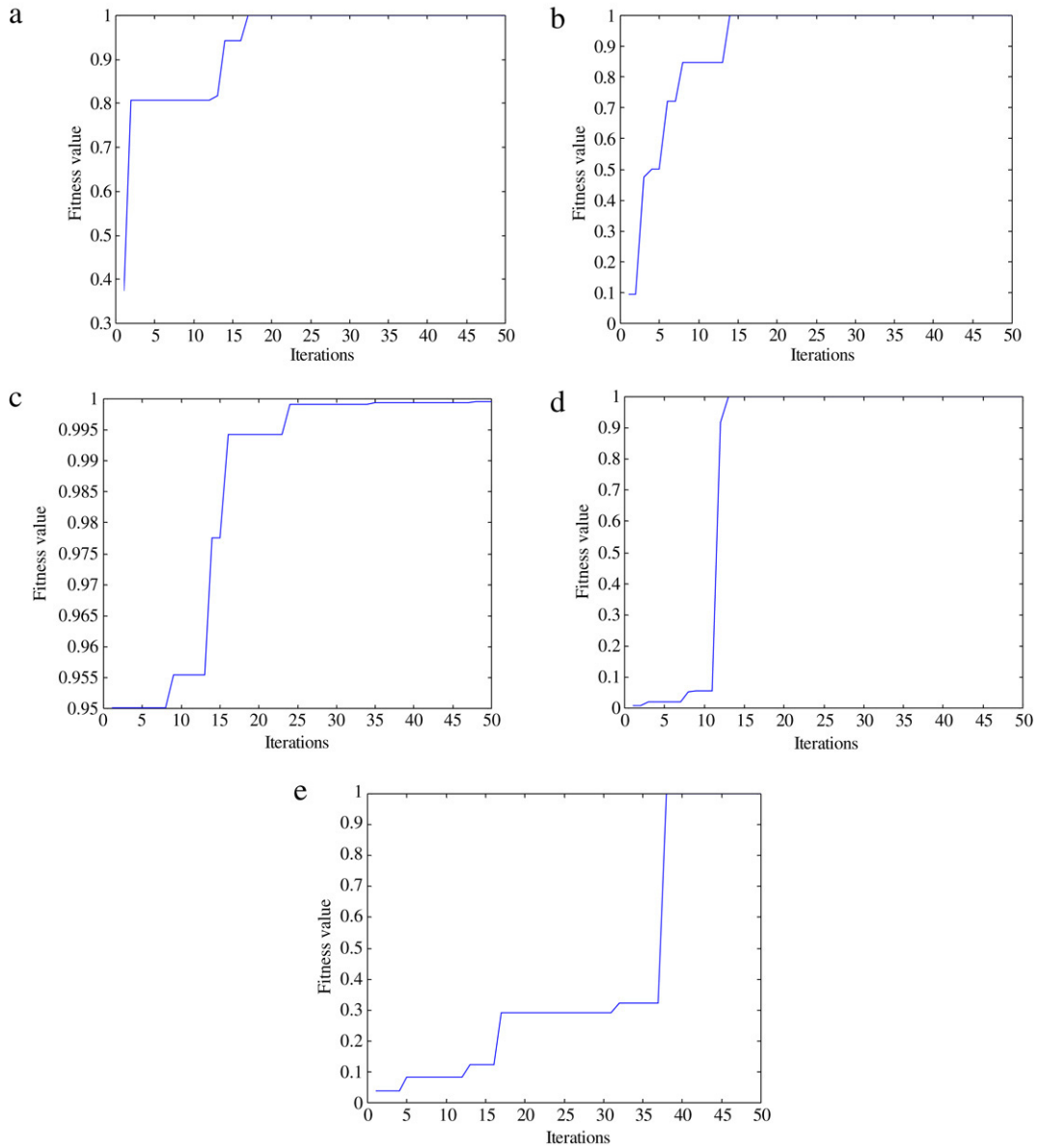


Fig. 8. Convergence of the fitness values through DNACA; (a) D_1 ; (b) D_2 ; (c) D_3 ; (d) D_4 ; (e) D_5 .

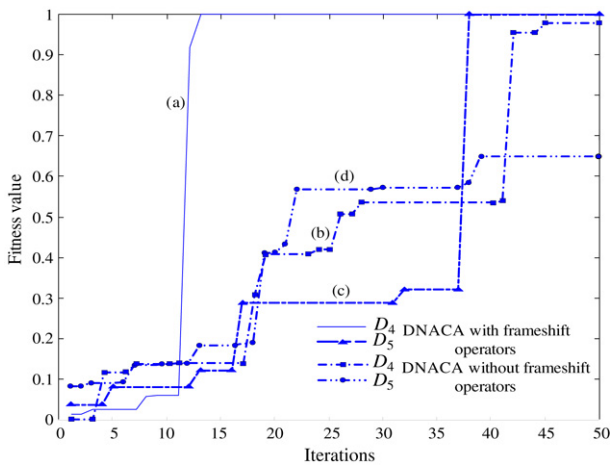


Fig. 9. Convergence of the fitness values for D_4 , D_5 through DNACA; (a), (c) with frameshift operators; (b), (d) without frameshift.

Our aim is to create a simplified transfer function to faithfully mimic the system behavior in the frequency domain. The frequency band under consideration is $[10^{-2}, 10^3]$ rad/s, four frequency intervals are considered individually which include $[10^{-2}, 1]$ rad/s, $[10^{-2}, 10]$ rad/s, $[10^{-2}, 10^2]$ rad/s and $[10^{-2}, 10^3]$ rad/s. For each section, DNACA is used to optimize the structures and parameters of the identified transfer function. Every section encompasses the result obtained in the previous section as the basis for the optimization process.

After 800 generations of evolution (i.e. every interval performed 200 generations of evolution), the final results are shown in Fig. 10(a)–(d) and is summarized in Table 3. For the first frequency interval, DNACA generates a transfer function approximating the low frequency behavior of P_0 . As it is displayed in Fig. 10(a), there exhibits a slight difference in the Bode magnitude and the phase plots of two systems. Clearly, the parameters and structure of the system model identified at this stage need to be refined. Moving further to the subsequent stages of evolution with the result obtained in the first stage as the initial model, the identification error attenuates with the increasing frequency interval of interest.

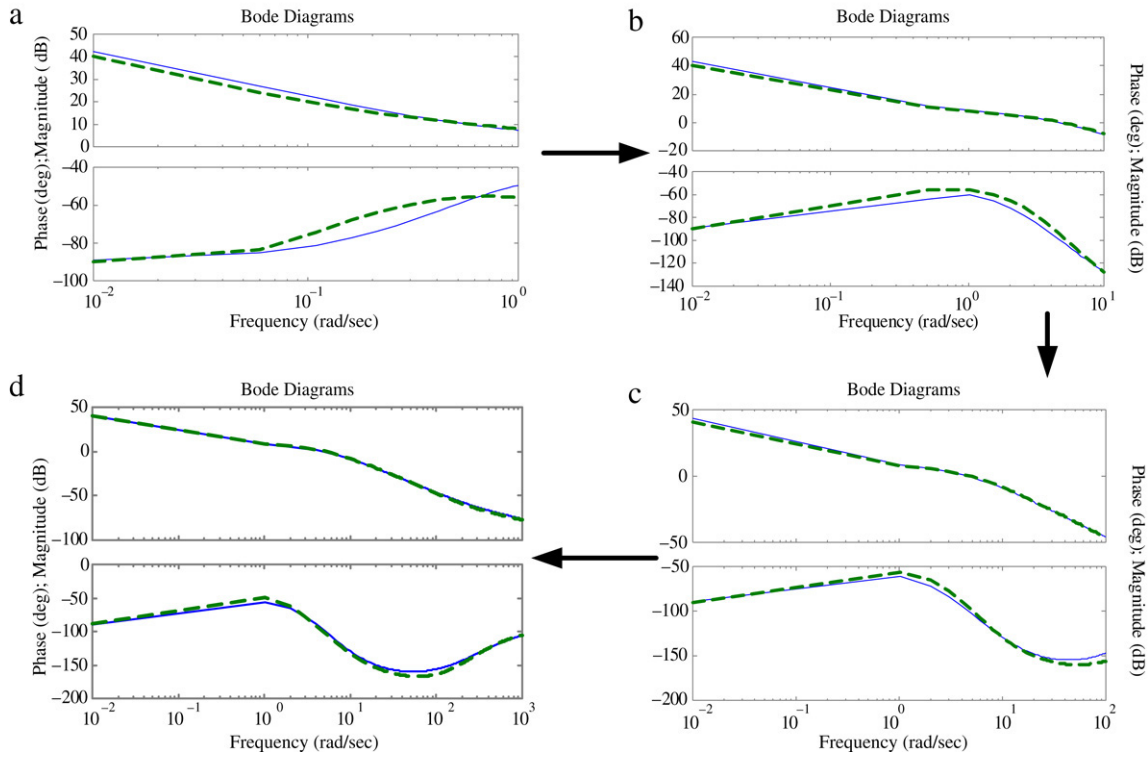


Fig. 10. Frequency response of identification on the interval from 10^{-2} to 10^3 : (a) Interval $[10^{-2}, 1]$; (b) Interval $[10^{-2}, 10]$; (c) Interval $[10^{-2}, 10^2]$; (d) Interval $[10^{-2}, 10^3]$.

Table 4
Results of estimation in different interval sections.

Estimation model	Performance indices		
	Max. tracking error (a)	Order of estimation model (b)	$\exp^{-((a) \times 0.1(b))}$
$\frac{1.9711(s + 5.674)(s + 9.403)(s + 18.45)}{s(s + 1.315)(s + 52.81)(s + 109.1)(s + 193.6)(s + 0.5288)(s + 54.45)(s + 62.29)(s + 69.79)}$ $10^{-2}-1$ rad/s	2.8030 dB(at 0.06 rad/s)	12	0.0346
$\frac{0.2332(s + 0.5593)(s + 192.1)}{s(s + 2.199)(s + 7.964)}$ $10^{-2}-10$ rad/s	2.8536 dB (at 0.01 rad/s)	5	0.2401
$\frac{0.2332(s + 0.5593)(s + 192.1)}{s(s + 2.199)(s + 7.964)}$ $10^{-2}-10^2$ rad/s	2.8536 dB (at 0.01 rad/s)	5	0.2401
$\frac{0.1355(s + 0.414)(s + 189.8)(s + 194.8)}{s(s + 6.236)(s + 3.081)(s + 103.1)}$ $10^{-2}-10^3$ rad/s	1.7315 dB (at 260.01 rad/s)	7	0.2975

Table 5
Coefficients of Hartman functions.

i	$a_{ij}, j = 1, \dots, 6$							c_i	$p_{ij}, j = 1, \dots, 6$						
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886		
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991		
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650		
4	17	0.05	8	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381		

The result reaches its optima over the whole frequency range after the fourth stage of identification; see Fig. 10(b)–(d). The summarized details are shown in Table 4.

To summarize these results, an estimation model with excellent identification accuracy is achievable through this DNACA. Moreover, through an appropriate fitness function setting and the introduction of two frameshift operators, the structure of the ID model could be reduced to its simplest form while simultaneously reaching an acceptable identification result.

Case 2. Comparison between CEP, FEP and DNACA

Consider the following test functions (Yao et al. [12]): Schwefel's Problem:

$$D_6 = \sum_{i=1}^5 |x_i| + \prod_{i=1}^5 |x_i|, \quad x_i \in [-10, 10], \quad f_{\min} = 0. \quad (20)$$

Generalized Griewank Function:

$$D_7 = \frac{1}{4000} \sum_{i=1}^5 x_i^2 - \prod_{i=1}^5 \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$

Table 6

Functions	Methods		
	FEP	CEP	DNACA
D_6	$8.1 \times 10^{-3}(2000)$	$8.1 \times 10^{-2}(2000)$	0(100)
D_7	$8.6 \times 10^{-4}(400)$	$8.1 \times 10^{-2}(400)$	$1.1 \times 10^{-3}(100)$
D_8	$5.3 \times 10^{-2}(1200)$	0.14(1200)	0(100)
D_9	$3.5 \times 10^{-7}(200)$	$3.8 \times 10^{-3}(200)$	$2.31 \times 10^{-7}(100)$
D_{10}	-1.03(100)	-1.03(100)	-1.03(100)
D_{11}	0.398(100)	0.398(100)	0.398(100)
D_{12}	3.02(100)	3.02(100)	3(50)
D_{13}	-3.86(100)	-3.86(100)	-3.86(100)
D_{14}	-3.27(100)	-3.28(100)	-3.301(200)

$$x_i \in [-600, 600], \quad f_{\min} = 0. \tag{21}$$

Ackley's Function:

$$D_8 = -20 \exp \left(-0.2 \sqrt{\frac{1}{5} \sum_{i=1}^5 x_i^2} \right) - \exp \left(\frac{1}{5} \sum_{i=1}^5 \cos 2\pi x_i \right) + 21, \quad x_i \in [-32, 32], \quad f_{\min} = 0. \tag{22}$$

Generalized Penalized Function:

$$D_9 = 0.1 \left\{ 10 \sin^2(3\pi x_1) + \sum_{i=1}^4 (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_5 - 1) (1 + \sin^2(2\pi x_5)) \right\} + \sum_{i=1}^5 u(x_i, 5, 100, 4), \tag{23}$$

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leq x_i \leq a, x_i \in [-50, 50] \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

$$f_{\min} = 0.$$

Six-Hump Camel-Back Function:

$$D_{10} = 4x_1^2 - 2.1x_2^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x_i \in [-5, 5], \quad f_{\min} = -1.0316 \tag{24}$$

Brain Function:

$$D_{11} = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10, \quad x_i \in [-5, 10] \times [0, 10], \quad f_{\min} = 0.398. \tag{25}$$

Goldstein-Price Function:

$$D_{12} = (1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) (30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)), \quad x_i \in [-2, 2], \quad f_{\min} = 3. \tag{26}$$

Hartman's Family Functions:

$$D_{13} = - \sum_{i=1}^4 c_i \exp \left(- \sum_{j=1}^3 a_{ij} (x_j - p_{ij})^2 \right), \quad x_i \in [0, 1], \quad f_{\min} = -3.86 \tag{27}$$

$$D_{14} = - \sum_{i=1}^4 c_i \exp \left(- \sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right), \quad x_i \in [0, 1], \quad f_{\min} = -3.32 \tag{28}$$

where a_{ij} and p_{ij} are shown in Table 5.

The fitness functions for $D_{6,7,8,9}$ are defined as

$$F_D^1 = \frac{1}{1 + D_i(x)}, \quad i = 6, 7, 8, 9 \tag{29}$$

The fitness functions for $D_{10,13,14}$ are defined as

$$F_D^2 = -D_j, \quad j = 10, 13, 14 \tag{30}$$

The fitness function for $D_{11,12}$ are defined as

$$F_D^3 = C_{\max} - D_k, \quad k = 11, 12 \tag{31}$$

where C_{\max} is a positive scalar and is set to be 1000.

For each test function, the effect of the number of iterations and population were considered to allow 200 and 100, respectively. The probability rates of crossover and mutation were 0.8 and 0.2, respectively. The probability rates of enzyme and virus were 0.4. The results of the solution are searched by using the proposed DNACA as summarized in Table 6.

5. Conclusions

A DNA computing algorithm has been proposed in this paper. Capability for seeking the solutions of the multiple variable functions is first confirmed by considering a class of De Jong's functions as the testing object. We also take some proposed test functions to compare between CEP and FEP to prove the method. The proposed DNACA has also been extended to deal with the system identification problem. Numerical experiments confirm its excellence for this problem. The algorithm's multiple mutation mechanism allows it to simplify the identification model's structure simultaneously while generating the optimal parameter set during the evolution process.

Acknowledgement

This research was sponsored by National Science Council, Taiwan, ROC under the Grant NSC94-2213-E005-013.

References

- [1] Adleman LM. Molecular computation of solutions to combinatorial problems. *Science* 1994;266:1021–4.
- [2] Maley CC. DNA computation: Theory, practice, and prospects. *Evolutionary Computation* 1998;6:201–29.
- [3] Forbes N. *Imitation of life: How biology is inspiring computing*. Cambridge: MIT Press; 2004.
- [4] Cosic I, Vojisavljevic V, Pavlovic M. The relationship of the resonant recognition model to effects of low-intensity light on cell growth. *International Journal of Radiation Biology* 1989;56:179–91.
- [5] De Jong KA. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation. Ann Arbor (MI): University Michigan; 1975.
- [6] Chen J, Antipov E, Lemieux B, Cedeno W, Wood DH. DNA computing implementing genetic algorithms. In: *Proceeding of DIMACS workshop on evolution as computation*. 1999. p. 39–49.
- [7] Jan HY, Lin CL, Hwang TS. Self-organized PID control design using DNA computing approach. *Journal of the Chinese Institute of Engineers* 2006;29(2): 251–61.
- [8] Tsuboi Y, Ibrahim Z, Ono O. DNA computing approach to knowledge representation. *International Journal of Hybrid Intelligent Systems* 2005;2: 1–12.
- [9] Goldberg DE. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston (MA): Kluwer Academic Publishers; 1989.
- [10] Hanaki Y, Hashiyama T, Okuma S. 1999. Accelerated evolutionary computation using fitness estimation. In: *Proceeding IEEE international conference on systems, man, and cybernations*. vol. 1. 1999. p. 643–8.
- [11] Frank HFL, Lam HK, Ling SH, Peter KST. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks* 2003;14:79–88.
- [12] Yao X, Liu Y, Lin G. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 1999;3(2):82–102.