International Journal of Swarm Intelligence and Evolutionary Computation

ISSN: 2090-4908

Sharif Ullah, Int J Swarm Intel Evol Comput 2014, 4:2
DOI: 10.4172/2090-4908.1000e111

Editorial Open Access

In Silico DNA computing

AMM Sharif Ullah*

Department of Mechanical Engineering, Kitami Institute of Technology, Japan

Introduction

To solve complex computational problems, researchers across all academic disciplines have been using the nature-inspired computing paradigms (swarm intelligence, evolutionarily computing, artificial neural network, DNA computing, and so on). This editorial deals with the *in silico* DNA [1-4].

DNA computing, one of the constituents of nature-inspired computing has two forms, namely, *in vitro* DNA computing and *in silico* DNA computing. *In vitro* DNA computing is the most popular one, where the authors deal with the topology of a dynamic structure consists of macromolecules (e.g., DNA) to solve computational problems. It started with the work of Adleman where *in vitro* DNA strands were synthesized to solve the travelling salesman problem [5]. Ullah et al. [4] have given an account of the usages of *in vitro* DNA computing. On the other hand, *in silico* DNA computing, the other form of DNA computing takes inspirations from the central dogma of molecular biology [6] and performs computation in silicon-based computing machines (i.e., in ordinary computers) to solve computational problems. The central dogma of molecular biology means "once (sequential) information has passed into protein it cannot get out again" [6]. In other words, mappings of DNA/RNA to DNA/RNA/protein are possible but not protein to DNA/RNA/protein [3,4].

One of the remarkable features of *in silico* DNA computing is that it can identify the similarity (or dissimilarly) of a set of objects even though the objects may look different (or similar) in terms of the usual appearances. For example, Ullah [3] has shown that the *in silico* DNA computing is powerful enough in distinguishing a normal pattern from an abnormal pattern exhibited by the stochastic time series called control chart used in statistical process control. In particular, it is shown that the *in silico* DBC does not produce any false alarm (does not misunderstand an abnormal pattern as a normal pattern) even though the window size (number of data points in the time series of control chart) is kept very short (less than and equal to 15). This is impossible to achieve if other nature-inspired computing paradigms (e.g., artificial neural network) are used to solve the same problem. Ullah et al. [4] have shown that the *in silico* DNA computing can be used to understand a shape from its stochastic variants, i.e., it can identify whether or not a set of images refers to an object irrespective of their visual differences. This article shed some lights on this issue (shape understanding ability of *in silico* DNA computing) using a numerical example. Before doing so, this study briefly describes the nature of *in silico* DNA computing.

Nature of *in silico* DNA computing

As mentioned before, *in silico* DNA computing takes inspiration from the central dogma of molecular biology and perform successive mappings: DNA to RNA and RNA to protein. Figure 1 schematically illustrates the *in silico* DNA computing that is intended for solving a computational problem. As seen from Figure 1, *in silico* DNA computing first maps a problem to a four-letter sequence called DNA strand consisting of the letter A, C, G, and T. On can create as many DNA strands as possible using the same piece of problem-relevant information. It then reorganizes the DNA strands to form another four-letter sequence called mRNA. Each three consecutive bases in the mRNA (known as codon, as shown in Table 1) are then mapped to a one-letter symbol of an amino acid to form the protein. To do this, the genetic rules are used as shown in Table 1 [3-4, 7]. The one-letter symbols of amino acids are A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, X, and Y. Note that "X" stands for the no-amino-acid situation (i.e., replaces the stop codons TAA, TAG, and TGA) [7]. The informational characteristics of the amino acids in the protein are used to extract some

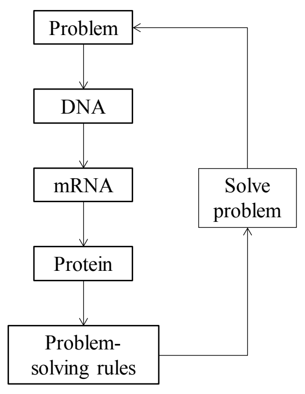

Figure 1: *In silico* DNA computing [3-4].

$m_j m_{j+1} m_{j+2}$	p_k	$m_j m_{j+1} m_{j+2}$	p_k	$m_j m_{j+1} m_{j+2}$	p_k	$m_j m_{j+1} m_{j+2}$	p_k
AAA	K	CAA	Q	GAA	E	TAA	X
AAC	N	CAC	H	GAC	D	TAC	Y
AAG	K	CAG	Q	GAG	E	TAG	X
AAT	N	CAT	H	GAT	D	TAT	Y
ACA	T	CCA	P	GCA	A	TCA	S
ACC	T	CCC	P	GCC	A	TCC	S
ACG	T	CCG	P	GCG	A	TCG	S
ACT	T	CCT	P	GCT	A	TCT	S
AGA	R	CGA	R	GGA	G	TGA	X
AGC	S	CGC	R	GGC	G	TGC	C
AGG	R	CGG	R	GGG	G	TGG	W
AGT	S	CGT	R	GGT	G	TGT	C
ATA	I	CTA	L	GTA	V	TTA	L
ATC	I	CTC	L	GTC	V	TTC	F
ATG	M	CTG	L	GTG	V	TTG	L
ATT	I	CTT	L	GTT	V	TTT	F

Table 1: Genetic codes [3-4,7].

***Corresponding author:** AMM Sharif Ullah, Department of Mechanical Engineering, Kitami Institute of Technology, Japan, Tel: + 81-157-26-9207; E-mail: ullah@mail.kitami-it.ac.jp

Received July 25, 2015; **Accepted** July 29, 2015; **Published** August 04, 2015

Citation: Sharif Ullah AMM (2015) *In Silico* DNA computing. Int J Swarm Intel Evol Comput 4: e111. doi: 10.4172/2090-4908.1000e111

rules for solving the problem. Here, informational characteristics of protein mean (entropy, absence, presence, abundance of some selected amino acids, and relationships among their likelihoods).

Let B = <...$b_i b_{i+1}$....> = <0100101011101100001000010> be the problem-relevant information, $\forall b_i \in \{0,1\}$. This piece of information can be converted into a DNA strand in different ways. For example, consider the following rules: $b_i b_{i+1} = 00 \rightarrow$ A, $b_i b_{i+1} = 01 \rightarrow$ C, $b_i b_{i+1} = 10 \rightarrow$ G, and $b_i b_{i+1} = 11 \rightarrow$ T. These rules transform B to a DNA strand denoted as DNA$_1$ = <...d_{1i}...> = <CGACGCGCTTGCTGAAACGAAACG>, $d_{(.)i} \in$ {A, C, G, T}. Consider another set of rules, as follows: $(b_i + b_{i+1} + b_{i+2}) = 0 \rightarrow$ A, $(b_i + b_{i+1} + b_{i+2}) = 1 \rightarrow$ C, $(b_i + b_{i+1} + b_{i+2}) = 2 \rightarrow$ G, $(b_i + b_{i+1} + b_{i+2}) > 2 \rightarrow$ T. If these rules are applied on the same problem-relevant information (B), then the following DNA strand denoted as DNA$_2$ = <...d_{2i}...> = <CCCCGCGGTGGGGCAACCCAACC> forms, instead. This way one can create a set of DNA strands from the same piece of problem-relevant information. Once a set of DNA strands are available, one can create an mRNA. In this case, too, a set of user-defined rules are needed. Let mRNA = <...m_j...> be the form of mRNA. Consider the following rule: $m_j m_{j+1} = d_{1i} d_{2i}$, i.e., two consecutive bases of mRNA come from two strands of DNA, DNA$_1$ and DNA$_2$, respectively. This yields mRNA = <CCGCACCCGGCCGGCGTTTGGGCGTGGCAAAAACCCGCAAAAACCC>. One may use other rules to form an mRNA from the strands of DNA, however. There is no restriction as such. It is worth mentioning that when there is only one strand of DNA, it serves the purpose of mRNA, i.e., it is directly used to create the sequence of amino acids (protein). Let Protein = <...p_k...> be the protein, The mapping $m_j m_{j+1} m_{j+2} \rightarrow p_k$ (Table 1) creates it (the protein) from the given mRNA = <...$m_j m_{j+1} m_{j+2}$...>. The mapping continues up to the last possible codon in the mRNA. The above mRNA yields Protein = < PHPAGVWAWQKPAKT>.

It is worth mentioning that the successive mappings underlying the *in silico* DNA computing (problem-relevant information to DNA, DNA to mRNA, and mRNA to protein, help gain information) increase the information content (entropy) in the system. The maximum possible information content of the problem-relevant information (B) is $\log_2(2) = 1$ Bit. The maximum possible information content of DNA/mRNA is $\log_2(4) = 2$ Bits. The maximum possible information content of protein is $\log_2(21) = 4.4$ Bits. The protein shown above (Protein = <PHPAGVWAWQKPAKT>) has nine different amino acids showing an entropy of 3.01 Bits, which is larger than that of DNA/mRNA. This increase in the information content due to the formation of many-element sequence from few-element sequence helps solve a problem, as shown in the next section.

Understanding Complex Shapes

Ullah et al. [4] have described how to use *in silico* DNA computing to understand complex shapes. Using the described procedure [4], a case study has been conducted to understand the two variants of an IFS fractal [8] taking the shape of a dragon. Figure 2 shows three variants of IFS dragon consisting of 100, 1000, and 5000 points. The shape is not clear from Figure 2a. Table 2 shows the frequencies of amino acids

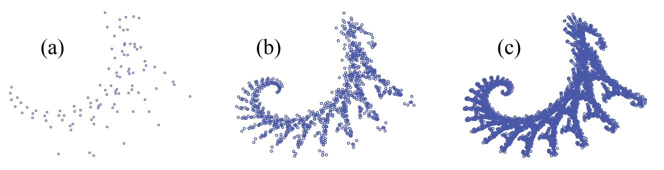

Figure 2: Three variants of a shape created by IFS fractal [8].

Amino Acids	Frequencies (*fr*(.))		
	(a)	(b)	(c)
A	4	126	260
C	**0**	**31**	**166**
D	**1**	**92**	**113**
E	**96**	**390**	**379**
F	0	10	638
G	**0**	**0**	**0**
H	**0**	**0**	**0**
I	**0**	**0**	**0**
K	18723	16816	15311
L	0	204	794
M	**0**	**0**	**0**
N	**96**	**390**	**379**
P	**0**	**0**	**0**
Q	**0**	**0**	**0**
R	101	447	266
S	**0**	**0**	**0**
T	**97**	**482**	**492**
V	**0**	**0**	**0**
W	**0**	**0**	**0**
X	**0**	**130**	**320**
Y	0	0	0

Table 2: Frequencies of amino acids in the proteins of the shape.

in the proteins of three variations of the shape. Needless to say, the proteins have been created using the in silico DNA computing defined in [4]. The *informational* characteristics of the frequencies of the amino acids are very similar. For example, consider the frequencies of amino acids denoted as E and N. The results shown in Table 2 confirm that $fr(\text{E}) = fr(\text{N})$ for all three cases. In addition, consider the frequencies of amino acids denoted as D, E, and T. The results shown in Table 2 confirm that $fr(\text{D}) + fr(\text{E}) = fr(\text{T})$ for all three cases. Moreover, consider the frequencies of G, H, I, M, P, Q, S, V, W, and Y. For all three cases, their frequencies are equal to zero. Consider the frequencies of C and X. Their frequencies gradually increase with the number of points representing the shape.

In synopsis, all three variants of a shape shown in Figure 2 possess the same information characteristics as carried by their proteins. In other words, if we study the informational characteristics of the proteins of the images shown in Figure 2, we can easily come to the conclusion that they refer to the same shape (dragon) even though the visual inspection may not support this argument as such.

Concluding Remarks

Unlike other nature-inspired computing paradigms, in silico DNA computing is still its infant state. A great deal of works lies ahead. The reader might explore in silico DNA computing from a different perspective and let this computing paradigm growth further in solving even more complex computing problems.

References

1. Ullah AMMS, Yano A, Higuchi M. (1997). Protein synthesis algorithm and new metaphor for selecting optimal tools. JSME International Journal, Series C, 40: 540–546.

2. Ullah AMMS (2003). Different facets of a computational equivalent of genetic addition. Biosystems. 68: 31–41.

3. Ullah AMMS (2010). A DNA-based computing method for solving control chart pattern recognition problems. CIRP Journal of Manufacturing Science and Technology 3: 293–303.

4. Ullah AMMS, D'Addona D, Arai N(2014) DNA based computing for understanding complex shapes, BioSystems 117: 40−53.
5. Adleman LM(1994). Molecular computation of solutions to combinatorial problems. Science, 266:5187:1021–1024.
6. Crick F (1970). Central dogma of molecular biology. Nature 227: 561–563.
7. UPAC-IUB Joint Commission on Biochemical Nomenclature (JCBN) (1984) Nomenclature and symbolism for amino acids and peptides, recommendations 1983. European Journal of Biochemistry, 138: 9–37.
8. Ullah AMMS, Sato Y, Kubo A, Tamaki J (2015) Design for manufacturing of IFS fractals from the perspective of Barnsley's fern-leaf, Computer-Aided Design and Applications, 12: 241−255.

Citation: Sharif Ullah AMM (2015) *In Silico* DNA computing. Int J Swarm Intel Evol Comput 4: e111. doi: 10.4172/2090-4908.1000e111