Available online at www.sciencedirect.com
ScienceDirect
BioSystems 92 (2008) 61–68
www.elsevier.com/locate/biosystems

Reconstruction of DNA sequences using genetic algorithms and cellular automata: Towards mutation prediction?

Ch. Mizas[a], G.Ch. Sirakoulis[a], V. Mardiris[a], I. Karafyllidis[a], N. Glykos[b], R. Sandaltzopoulos[b,*]

[a] *Democritus University of Thrace, Department of Electrical and Computer Engineering, 67100 Xanthi, Greece*
[b] *Democritus University of Thrace, Department of Molecular Biology and Genetics, Laboratory of Gene Expression, Molecular Diagnostics and Modern Therapeutics, 68100 Alexandroupolis, Greece*

Received 21 December 2006; received in revised form 28 November 2007; accepted 10 December 2007

Abstract

Change of DNA sequence that fuels evolution is, to a certain extent, a deterministic process because mutagenesis does not occur in an absolutely random manner. So far, it has not been possible to decipher the rules that govern DNA sequence evolution due to the extreme complexity of the entire process. In our attempt to approach this issue we focus solely on the mechanisms of mutagenesis and deliberately disregard the role of natural selection. Hence, in this analysis, evolution refers to the accumulation of genetic alterations that originate from mutations and are transmitted through generations without being subjected to natural selection. We have developed a software tool that allows modelling of a DNA sequence as a one-dimensional cellular automaton (CA) with four states per cell which correspond to the four DNA bases, i.e. A, C, T and G. The four states are represented by numbers of the quaternary number system. Moreover, we have developed genetic algorithms (GAs) in order to determine the rules of CA evolution that simulate the DNA evolution process. Linear evolution rules were considered and square matrices were used to represent them. If DNA sequences of different evolution steps are available, our approach allows the determination of the underlying evolution rule(s). Conversely, once the evolution rules are deciphered, our tool may reconstruct the DNA sequence in any previous evolution step for which the exact sequence information was unknown. The developed tool may be used to test various parameters that could influence evolution. We describe a paradigm relying on the assumption that mutagenesis is governed by a near-neighbour-dependent mechanism. Based on the satisfactory performance of our system in the deliberately simplified example, we propose that our approach could offer a starting point for future attempts to understand the mechanisms that govern evolution. The developed software is open-source and has a user-friendly graphical input interface.

Keywords: Genetic algorithms; Cellular automata; DNA sequence modelling; Evolution; Simulation; Bioinformatics; Nanotechnology

1. Introduction

In the recent years, advances of biochemical and biophysical instrumentation methodologies allowed collection of whole genome and proteome sequences (Venter et al., 2001). The complexity of the available information calls for the development of ever more powerful bioinformatics tools to attack complex problems associated with proper interpretation of the obtained information. Faster and more accurate computational intelligence algorithms and data processing technologies are required (Baxevanis and Ouellette, 1998; Chou and Zhang, 1992; Cios et al., 2005; Durbin et al., 2000; Zhang and Chou, 1993, 1994; Sirakoulis et al., 2004). In general, computational intelligence methods may be employed provided that they are wisely combined with bioinformatics, as illustrated in Baxevanis and Ouellette (1998). In order to contribute to this effort we modelled DNA sequence as a one-dimensional cellular automaton (CA) thus allowing the use of powerful and sophisticated computation techniques for the development of useful bioinformatics tools (Sirakoulis et al., 2003).

CAs, originally developed by von Neumann (1966) as models of self-reproducing systems, have been extensively used to model and simulate environmental and biological systems (Bernadres and dos Santos, 1997; Chou et al., 2006; Gao et al., 2006; Gaylord and Nishidate, 1996; Kansal et al., 2000; Karafyllidis, 1997, 1998; Karafyllidis and Thanailakis, 1997; Patel et al., 2001; Salzberg et al., 2004; Sirakoulis et al., 2000; Wang et al., 2005; Xiao et al., 2005a,b, 2006a,b). CAs were

* Corresponding author. Tel.: +30 25510 30622; fax: +30 25510 30624.
E-mail address: rmsandal@mbg.duth.gr (R. Sandaltzopoulos).

0303-2647/$ – see front matter © 2008 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2007.12.002

showed to be a promising model for DNA sequence (Sirakoulis et al., 2003), because certain aspects of the DNA structure, function and evolution can be simulated using several mathematical tools (such as linear algebra and operators), introduced through the use of CAs. In our model the sugar-phosphate backbone of a DNA molecule corresponds to the CA lattice and the organic bases to the CA cells. At each position of the lattice one of the four bases A (adenine), C (cytosine), T (thymine) and G (guanine) of the DNA molecule may be allocated. These four bases correspond to the four possible states of the CA cell.

In elementary CAs, the CA evolution rule can be extracted from a given number of CA evolution patterns. This method can also be applied to the CAs that model DNA sequences. Thus, we developed a simulator, named DNA_EVO, which allows modelling DNA evolution by extracting CA rule(s) using genetic algorithms (GAs) (Holland, 1975; Goldberg, 1989). The evolution rule(s) can be determined by providing the global state of the DNA sequence in various evolution steps. Conversely, if the evolution rule(s) and DNA sequences are given for several evolution steps, it may be possible to determine the DNA sequence in previous evolution steps. DNA_EVO is an interactive simulation tool that includes a graphical user interface [GUI] which has been implemented using Tcl/Tk facilities (Welch et al., 2003).

2. Modelling DNA in Terms of Cellular Automata

In this work we match certain properties of DNA with the features of a proposed CA thus modelling DNA in terms of CAs. Sketching certain directions that DNA modelling might take, we hope to stimulate further research in this direction.

Cellular automata (CAs) were introduced by von Neumann (1966) and Ulam (1974) as a possible idealization of biological systems, in order to model biological self-reproduction. In this work, we model DNA as a one-dimensional CA and, therefore, only one-dimensional CAs will be presented in this section (Sirakoulis et al., 2003). A one-dimensional CA consists of a regular uniform lattice, which may be infinite in size and expands in a one-dimensional space. Each site of this lattice is called a *cell*. At each cell a variable takes values from a discrete set. The value of this variable is the *state* of the cell. Fig. 1(a) shows a one-dimensional CA. The CA lattice consists of identical cells, $\ldots, i-3, i-2, i-1, i, i+1, i+2, i+3, \ldots$, and the corresponding states of these cells are C_{i-3}, C_{i-2}, C_{i-1}, C_i, C_{i+1}, C_{i+2} and C_{i+3}.

The state of the ith cell takes values from a predefined discrete set:

$$C_i \in \{c_1, c_2, c_3, \ldots, c_n\} \tag{1}$$

where $c_1, c_2, c_3, \ldots, c_n$ are the elements of the set. This set may be a set of integers, a set of real numbers, a set of atoms, a set of molecules, or even a set of properties. If this set contains only the two binary numbers, i.e. $C_i \in \{0, 1\}$, the CA is called *elementary*.

The CA is a dynamic system, which *evolves* in discrete time steps. Its evolution is manifested as a change of its cell states with time. The state of each cell is affected by the states of its neighbouring cells. All the cells that may possibly affect the change of the state of the ith cell are the *neighbourhood* of this cell. The neighbourhood is defined as follows:

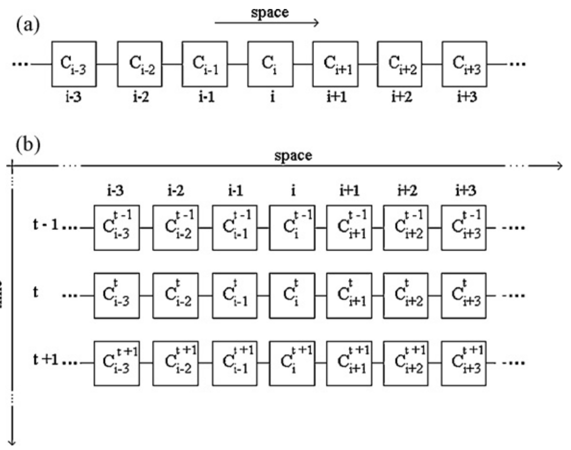

(a)
space
C_{i-3} C_{i-2} C_{i-1} C_i C_{i+1} C_{i+2} C_{i+3}
i-3 i-2 i-1 i i+1 i+2 i+3
(b)
space
time
t-1
t
t+1
C^{t-1}_{i-3} C^{t}_{i-3} C^{t+1}_{i-3}

Fig. 1. (a) An example of a typical one-dimensional CA; (b) a sequence of evolution steps of a one-dimensional CA.

$$N(i, r) = \{C_{i-r}, \ldots, C_{i-3}, C_{i-2}, C_{i-1}, C_i, C_{i+1}, C_{i+2}, C_{i+3}, \ldots, C_{i+r}\}, \quad r = 0, 1, 2, 3, \ldots, m \tag{2}$$

where r is the size of the neighbourhood. If $r = 1$, which is the most usual case then the neighbourhood of the ith cell consists of the same cell and its left and right immediate neighbours:

$$N(i, 1) = \{C_{i-1}, C_i, C_{i+1}\} \tag{3}$$

The state of the ith cell at time step $t+1$ is affected by the states of its neighbours at the previous time step t, i.e. the state of the ith cell at a time step is a function of the states of its neighbours at the previous time step:

$$C_i^{t+1} = F(C^t_{i-r}, \ldots, C^t_{i-3}, C^t_{i-2}, C^t_{i-1}, C^t_i, C^t_{i+1}, C^t_{i+2}, C^t_{i+3}, \ldots, C^t_{i+r}) \tag{4}$$

This function is the CA *evolution rule*. The upper index in the state symbol denotes the time step. C_i^{t+1} is the state of the ith cell at time step $t+1$. If $r = 1$, Eq. (4) becomes

$$C_i^{t+1} = F(C^t_{i-1}, C^t_i, C^t_{i-1}) \tag{5}$$

Fig. 1(b) shows the evolution of a one-dimensional CA. The horizontal axis is *space* and the vertical axis is *time*. Each row represents the CA at each time step and each column represents the state of the same cell at various time steps.

DNA may be modelled as a one-dimensional CA; the sugar-phosphate backbone corresponds to the CA lattice where the four bases A, C, T and G may be attached, each one corresponding to the four possible states of a CA cell. The state of the ith cell of this CA takes values from the discrete set that comprises the four bases:

$$C_i \in \{A, C, T, G\} \tag{6}$$

We define as an *evolution event* a change in the state of one or more CA cells. Therefore, any point mutation (i.e. a mutation of a single nucleotide) is an evolution event and it corresponds to a cell state change. If a DNA strand is passed unaltered from one generation to the other, no state change occurs, and the CA does not evolve. The CA evolves if there occurs at least one change in one of its cells and this change is transmitted to an offspring.

A time step in CA evolution is determined as the time interval between any pair of subsequent CA cell changes. So, the time flow is not uniform because mutations do not occur in regular intervals. Consider for example an asexually reproducing species with a generation period of 1 day. Suppose that a CA cell state change (DNA mutation) occurs now, the next one occurs in 10 days, the next one in 3 days and the next one in 6 days. Then the first time step represents 10 days of real time, the second 3 days and the third 6 days. But, in the CA model all time steps are equivalent, i.e. the difference in real time between the first, the second and the third time step does not become evident. An advantage of modelling DNA sequence as a CA is that the DNA strand and the individuals transmitting it from one generation to the other may exist in different time scales and, therefore, the lifespan and the generation period of the individuals that bear it appear to be time-like irrelevant for DNA evolution.

Since CAs are deterministic computational models, their usefulness in modelling DNA evolution is proportional to the degree that DNA evolution is a deterministic procedure. Indeed there are indications that the mechanisms of mutagenesis are not absolutely random procedures (McFadden and Al-Khalili, 1999; Schwefel, 2002); for example, neighbour-dependent mutation has been studied using Markov chains and revealed biases in mutation rates that depend on the neighbouring bases (Arndt et al., 2001). Therefore, we will proceed to the construction of our model by assuming that mutations, i.e. CA cell changes, depend on the states of some of the cells that are located nearby.

Suppose that a state change at the *i*th cell occurs, and a time step is taken. In the model presented here it is supposed that the state of this cell has changed under the influence of the states of its neighbours. The new state of the *i*th cell at this time step (which is generally the $t+1$ step) is given by

$$C_i^{t+1} = \hat{M}(C_{i-r}^t, \ldots, C_{i-3}^t, C_{i-2}^t, C_{i-1}^t, C_i^t, C_{i+1}^t, C_{i+2}^t, C_{i+3}^t, \ldots, C_{i+r}^t) \quad (7)$$

Eq. (7) is a more general expression of the evolution rule given in Eq. (4), where the function F has been replaced by an operator, $\hat{M}$, which is a more general mathematical abstraction. An operator may be a mathematical function, a logic function, a matrix, etc. The operator operates on the state of the neighbourhood of the *i*th cell at time step t and produces the state of this cell at time step $t+1$.

In Eq. (7) cell states correspond to one of the four bases A, C, T and G. Operators act on numbers and symbols that represent numbers. Therefore, the four bases must be represented by numbers. Since there are only four bases, the most appropriate way of representing them by numbers is to correspond each one of them to a respective number of a quaternary number system, comprising only four numbers, i.e. 0, 1, 2 and 3. We represent the bases with numbers as follows:

$$A \to 0, \quad C \to 1, \quad T \to 2, \quad G \to 3 \quad (8)$$

A vast number of evolution rules can be applied to the CA that models DNA. Interestingly, evolution rules that include base insertion *and/or* base deletion may be used. Usually, whenever a novel CA model is proposed, one initially studies linear evolution rules. The study of linear rules reveals the dynamics of the CA evolution and provides a very good insight to the structures created by evolution. The use of linear rules in our case is further justified by the fact that a linear algebra has already been successfully applied for the analysis of mutation rates (Jones et al., 1999).

In the case of linear evolution rules, the operator $\hat{M}$ of Eq. (7) is a matrix, M, and the evolution rule takes the form:

$$\begin{bmatrix} \vdots \\ C_{i-2}^{t+1} \\ C_{i-1}^{t+1} \\ C_i^{t+1} \\ C_{i+1}^{t+1} \\ C_{i+2}^{t+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & M_{i-2,j-2} & M_{i-2,j-1} & M_{i-2,j} & M_{i-2,j+1} & M_{i-2,j+2} & \cdots \\ \cdots & M_{i-1,j-2} & M_{i-1,j-1} & M_{i-1,j} & M_{i-1,j+1} & M_{i-1,j+2} & \cdots \\ \cdots & M_{i,j-2} & M_{i,j-1} & M_{i,j} & M_{i,j+1} & M_{i,j+2} & \cdots \\ \cdots & M_{i+1,j-2} & M_{i+1,j-1} & M_{i+1,j} & M_{i+1,j+1} & M_{i+1,j+2} & \cdots \\ \cdots & M_{i+2,j-2} & M_{i+2,j-1} & M_{i+2,j} & M_{i+2,j+1} & M_{i+2,j+2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ C_{i-2}^t \\ C_{i-1}^t \\ C_i^t \\ C_{i+1}^t \\ C_{i+2}^t \\ \vdots \end{bmatrix} \quad (9)$$

The column matrix at the right hand side of Eq. (9) is formed by the states of all CA cells at time step t. This matrix is multiplied by the matrix M, which represents the evolution rule. The matrix elements $M_{i,j}$ may take only two values, namely 0 and 1. The column matrix at the left hand side of Eq. (9) is the result of the matrix multiplication and it contains the states of all CA cells at time step $t+1$. All the additions are modulo 4 additions. In the case of a CA with n cells (DNA strand with n bases) the column matrices have n rows and the matrix M is a square matrix with n columns and n rows. Each square matrix M represents a CA rule.

In the case of elementary CAs, it has been shown that, given an evolution pattern, the evolution rule that generated it can be determined (Adamatzky, 1994). Similarly, if the DNA sequence at various time steps of a line of evolution is available, it may be possible to determine the evolution rule (or rules) that triggered this evolution line. Once the evolution rule and the DNA sequence at present time are known, it may be possible to predict the next evolution event (or events) and, therefore, the DNA sequence at the next time step.

3. Simulation of DNA Sequence Evolution Using the Proposed Model

We employed the model described above to simulate the evolution of DNA sequences. We have chosen to apply an evolution rule that incorporates only nearest neighbour interactions in our simulations, as most of the studies on mathematical models of DNA are limited to such interactions (Archilla et al., 2002). Hence, an evolution rule may be represented by the following matrix:

$$M=\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 1 & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 1 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 1 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 0 & 1 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \qquad (10)$$

All the elements in a matrix row are zero, except the three neighbouring elements that are equal to one. If this matrix is multiplied by the column matrix formed by the states of all CA cells, at time step t, the state of the ith element at time step $t+1$ will be the modulo 4 addition of its own state and the states of its left and right neighbours (cells $i-1$ and $i+1$, respectively), at time step t.

Fig. 2 shows the simulated evolution of a DNA sequence. The simulation starts with a random sequence of a DNA strand comprising 30 bases and produces the strands for 30 successive time steps. Base A is shown in white, C in light gray, T in dark gray and G in black.

A vast number of evolution rules can be applied to the presented CA that models DNA evolution. The CA rule space comprises every possible local rule that may be applied to the CA cells. In this simulation we have chosen to limit the mathematical model to the nearest neighbour. For CAs with only two states per cell, the number of possible rules is given by 2^{2^n}, where n is the number of cells in the neighbourhood. In one-dimensional

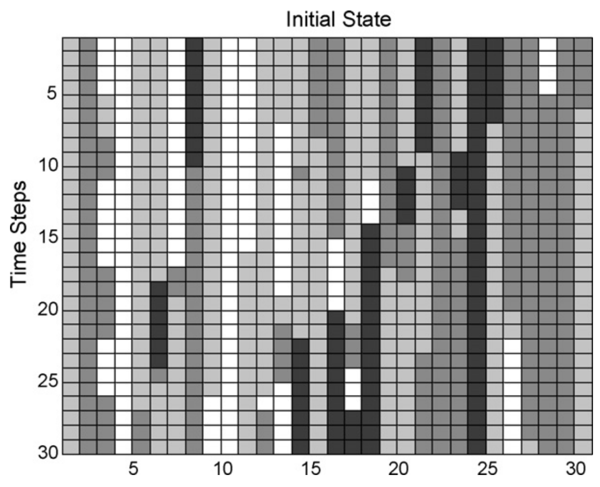


Fig. 2. Simulated evolution of a random DNA sequence: A, white; C, dark gray; T, light gray; G, black.

333	...	013	012	011	010	003	002	001	000	Possible States
r_{63}	...	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0	Rule Components
4^{63}	...	4^7	4^6	4^5	4^4	4^3	4^2	4^1	4^0	Coefficients

Fig. 3. Evolution rules of the four state classical CA model of DNA evolution.

CAs with only two states per cell, the neighbourhood of a cell comprises the cell itself and those on its left and right sides, hence the number of all possible rules is 2^8 (von Neumann, 1966), while in the four-state CA these rules are extended to 4^{4^3} or 4^{64}. The entire rule space of such CAs must be searched in order to find the putative CA evolution rules that govern the DNA sequence evolution. In this work GAs are utilized in order to explore the vast CA rule space. A possible evolution scheme of the proposed CA is shown in Fig. 3, where the first row gives the possible states the cells within the neighbourhood could take. The r_i's in the second row are the rule components which take values from the discrete set $\{0, 1, 2, 3\}$. The last row shows the coefficients associated with the corresponding components. The rule R can be defined as $R=(r_0, r_1, r_2, r_3, \ldots, r_{63})$. The numerical label D assigned to R is given by: $D=\sum_{s=0}^{2^6-1} r_s 2^s$, which is the sum of the coefficients associated with all nonzero components.

GAs have already been used to search the rule spaces of one-dimensional CAs with two states per cell, the neighbourhood of which comprises the cell itself and the three cells to the right and to the left of it (Das et al., 1994; Iovine et al., 2005; Karafyllidis, 2004; Mitchel et al., 1994; Yang and Billings, 2000). Such GAs have been utilized to design (i.e. to find the proper rule of) CAs that can perform two computational tasks: density classification and synchronization (Das et al., 1994; Iovine et al., 2005; Karafyllidis, 2004; Mitchel et al., 1994; Yang and Billings, 2000). Here we describe a GA that is suitable for the analysis of DNA sequence evolution and elaborate on its function.

GAs are search procedures that mimic the mechanics of genetics and natural selection. They provide robust searching capabilities in complex problem solution spaces. Possible solutions of an optimization problem are represented as a finite length string: $\{x_1, x_2, \ldots, x_n\}$, over an alphabet of finite length: $\{a_1, a_2, \ldots, a_m\}$. A possible solution represented in this way is called a "chromosome". Each chromosome contains smaller building blocks x_i, $i=1, 2, 3, \ldots, n$, referred to as "genes". A set of chromosomes is referred to as a "population". A target function (whose value we seek to optimize) provides the mechanism for evaluating the solution represented by each chromosome. This function is referred to as a "fitness function", and the value assigned to each chromosome, using the fitness function, is the "chromosome fitness". The chromosome with the highest fitness represents the optimal solution. Chromosomes of a relatively high fitness value represent very good or superior solutions, whereas chromosomes of a relatively lower fitness value represent bad or inferior solutions to the problem at hand (Holland, 1975).

Usually the alphabet used is $\{0, 1\}$ and chromosomes are strings of 0s and 1s: $\{1\,0\,0\,1 \ldots 0\,1\}$. In this case, each gene can be either 0 or 1. If the length of a chromosome is L, it is evident that the number of possible chromosomes is 2^L, each chromosome representing a different solution to the problem at hand. Within the entire population, the chromosome that represents the optimum, or a chromosome that represents a very good solution, has to be found. The entire population, i.e. the set of all possible chromosomes, is referred to as the “search space”. In most cases the search space is so large that conventional search techniques are ineffective. Therefore we employ GAs in order to search such a vast search space as in the case of DNA sequence analysis described above, aiming to discover the optimal, or a near-optimal, solution.

A GA starts with a randomly chosen initial population. The initial population comprises a relatively small number of chromosomes. Typically, GAs use an initial population size of 100–10,000. The fitness function is used to calculate the fitness of each chromosome in the initial population.

GAs produce the next generation by performing three genetic operations on the initial population, namely selection, crossover and mutation. It is noteworthy that the terms “selection”, “crossover” and “mutation” bear the meaning commonly referred to by computer scientists and are distinct from the terms used by population geneticists. Selection is a process that selects superior chromosomes, i.e. those with the most optimal fitness function values, which will survive to the next generation, and also selects inferior chromosomes, which will perish. A simple selection strategy is to allow a number of the fittest chromosomes to survive, and to discard the less fit ones.

In every generation, crossover generates offspring by exchanging genetic material between pairs of highly fitted chromosomes. We utilize crossover operation in order to introduce novel combinations of the genetic material. In this way it is likely to lead the evolution of the GA to the global optimum of its fitness function. Assuming that L is the chromosome length, a crossover point is chosen at random. The crossover point can assume values in the range 1 to $(L-1)$. The portions of the two chromosomes beyond this crossover point are exchanged to form two new chromosomes of the next generation. The crossover rate C controls the frequency with which the crossover operation is applied. If N is the size of the population $C \times N$ chromosomes undergo crossover in each generation.

After crossover, chromosomes are subjected to mutation. A random gene of a random chromosome is selected and mutated. Mutation of a gene changes its value from 0 to 1, or vice versa. Mutation operation increases the population variability thus helping to prevent irrecoverable loss of potentially important information concerning the solutions of the problem at hand. The mutation operation plays a secondary role in the creation of the new generation, and the mutation rate M is kept low. If N is the population size and L the chromosome length, then $M \times N \times L$ genes undergo mutation per generation.

The fitness of each chromosome is calculated using the fitness function, and a new generation is created by selection, crossover and mutation. The iteration stops when an arbitrarily acceptable solution is reached or after a given number of generations (Fogel, 1995).

The successful description of DNA evolution by a CA system, such as the one presented above, combined with the use of GA searches provide an extremely powerful simulation method to study DNA evolution.

4. DNA_EVO Simulation Tool

DNA_EVO is an automated simulation tool, based on the usage of GAs, designed to extract very efficiently the CA evolution rule, or rules that govern the evolution of DNA sequence. DNA_EVO's user interface has been implemented using Tcl/Tk GUI's facilities, enabling interactive simulation (Welch et al., 2003). The proposed tool is open-source and platform independent. No previous knowledge of CAs, GAs or computer programming is necessary to use the simulator, because of the user-friendly graphical user interface that has been developed.

After the assignment of the origin DNA sequence by the user, an initial population P that contains n possible solutions, meaning n CA evolution rules is constructed randomly. The value of n is user-defined and should be a compromise between accuracy and the availability of computer time and memory. For each possible solution i of population P with n individuals an error function is given by

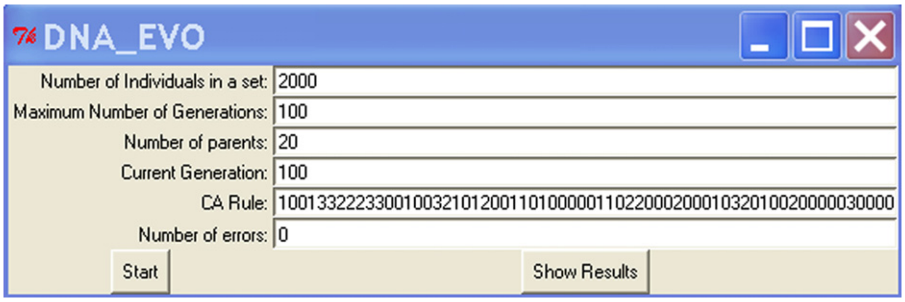
$$\mathrm{Mer}(i) = \sum_{j=0}^{\mathrm{SET}} |y(i, j) - \hat{y}(i, j)| \tag{11}$$

where $y(i, j)$ is the measured state at data point j for chromosome i and $\hat{y}(i, j)$ is the predicted state, in correspondence. Each chromosome in the current population is ranked with respect to ‘Mer’ of Eq. (11). The chromosome with the lowest ‘Mer’ value occupies the first position, the chromosome with the second lowest ‘Mer’ value occupies the second position and so on. Chromosomes with the same error share the same rank. After the final ranking we calculate the fitness function for each chromosome. The fitness function of the ith chromosome is defined as

$$\mathrm{fit}(i) = \frac{\mathrm{MAX}(\mathrm{rank}(i)) - \mathrm{rank}(i)}{\mathrm{MAX}(\mathrm{rank}(i)) - \mathrm{MIN}(\mathrm{rank}(i))} \tag{12}$$

The GA algorithm for the selection of CA evolution rules can be summarized as follows:

(1) Set the current generation number $i = 1$.
(2) Set the GA algorithm parameters.
(3) Generate a random population set P with n individuals.
(4) Compute ‘Mer’ (modulus of error function) for each individual in P (Eq. (11)).
(5) Rank the individual in R.
(6) Calculate the fitness function value of each chromosome (Eq. (12)).
(7) Apply the parent selection technique to P.
(8) Employ crossover and mutation to P to produce the corresponding offspring set P'.



Number of Individuals in a set: 2000
Maximum Number of Generations: 100
Number of parents: 20
Current Generation: 100
CA Rule: 100133222330010032101200110100000110220002000103201002000030000
Number of errors: 0

Fig. 4. The final screen of the DNA_EVO after the execution of the GA algorithm, showing the user defined input data and the extracted evolution rule. Note that there were no errors in this example, i.e. the defined evolution rule could be used successfully to reconstruct the final sequence.

(9) Calculate the corresponding fitness function for the chromosomes in the offspring set P'. Select the n fittest individuals from both the population set P and the corresponding set P', by comparing their fitness value. Reset P using the corresponding newly selected n individuals and nullify the offspring set P'.
(10) Set the generation number $i = i + 1$.
(11) Return to (4) and repeat until a pre-specified number of generations is reached.

The code of the GA algorithm was developed with the usage of Tcl/Tk and interacts in real time with the user's defined parameters. These parameters are the original DNA sequences, the DNA sequences of intermediate evolution steps, the final DNA sequences, the number of individuals in a set, the maximum number of generation and the number of parents. The aforementioned parameters remain unchanged during the simulation. It should be mentioned that DNA_EVO does not impose any *a priori* limitations on these parameters which are modifiable and can be changed interactively. Simulation results are saved in txt format suitable for numerical experiments and can be visualized in a graphical format with the help of the "Show results" button. Furthermore, the software verifies input data and, in case of parameter values that are unacceptable, the user is forced to correct these values.

A paradigm of the functional operation of DNA_EVO is presented in Figs. 4 and 5. In this example, the user defined the number of individuals to be equal to 2000, the maximum number of generations equal to 100 and the number of parents equal to 20. She/he has also inserted the DNA sequences with 100 bases length corresponding to the origin, the final and one intermediate evolution step, shown in Fig. 4 (upper part). In this example, sequences differing from each other by as much as ∼80% were used deliberately in order to provide an extremely hard test case. In the lower part of Fig. 4 are shown the DNA_EVO results after 100 generations in the DNA sequences, produced by the execution of the analogous CA rule. The CA rule found is shown in Fig. 4. It is obvious that DNA_EVO managed to reconstruct successfully the evolution pattern of the given DNA sequences without any errors. Apparently, evolution data visualization is straightforward, and the evolution patterns can be easily studied and interpreted.

Furthermore, the simulator was tested in 20 different cases, each one referred to two random 100 bases long DNA sequences, corresponding to the origin and the final evolution step, differing from each other by as much as ∼25%. Additionally, the number of individuals was chosen to be equal to 2000, the maximum number of generations equal to 200 and the number of parents equal to 100. In most of the cases (60%, data not shown) the simulator performed superbly and was able to decipher the CA rule without any error.

5. Discussion

We have modelled DNA as a one-dimensional cellular automaton where the sugar-phosphate backbone corresponds to the CA lattice and the organic bases to the CA cells. Every cell may have any of four possible states, corresponding to the four DNA bases A, C, T and G. These four states are represented by the quaternary number system. Linear evolution rules, represented by square matrices, were considered. In our model we assumed that DNA mutagenesis is influenced by the identity of the nucleotide to be mutated and the identity of the nucleotides in its vicinity. Based on this assumption, we developed proper GAs

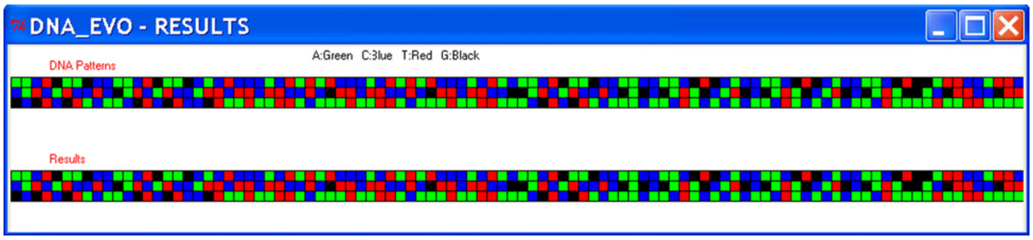


Fig. 5. The DNA sequences corresponding to the origin, the final and one intermediate evolution step provided by the user are shown on the upper panel. The sequences produced by the evolution process according to the CA rule defined by DNA_EVO are presented on the lower panel. It is striking that there is no mismatch between the corresponding sequences of these two panels proving the efficiency of the algorithm.

that efficiently extract the CA rule that governs sequence evolution. We reported the design of a software package, named DNA_EVO, that simulates DNA sequence evolution according to our model using a GA methodology for determining the evolution rule generating given evolution patterns.

The simulation tool has been tested successfully in random numerical experiments. In these tests, several time steps of the evolution of a DNA sequence were given and the simulator determined the possible rule (or rules) that generated the given evolution pattern. Hence, we showed that, given a set of DNA sequences that represent a series of evolution steps, the simulator can be used in reconstructing DNA sequences of missing evolution steps. If a series of snapshots of the evolutionary history of a DNA sequence is available, DNA_EVO can be used to make a plausible prediction about the subsequent evolution step, i.e. the next mutation that may take place.

Our model relies on the assumption that mutations are deterministic events and their evolution can be modelled using CAs. Although it is generally accepted that mutations happen at random, there are observations that challenge this idea. For example, a recent study using Markov chains revealed biases in mutation rates that depend on the neighbouring bases indicating a neighbour-dependent influence in the process of mutagenesis (Arndt et al., 2001). Recent quantum-mechanical models of DNA evolution have proposed that mutagenesis is directed by quantum-mechanical mechanisms (McFadden and Al-Khalili, 1999; Baake et al., 1997, 1998; Bieberich, 2000; Kirby, 2002; Ogryzko, 1997). Such models are strongly supported by recent data indicating that quantum proton tunneling causes tautomeric transitions in base pairs resulting in mutations during DNA replication (Golo and Volkov, 2002; Hjort and Stafstrom, 2001; Kryachko, 2002). In addition, there is at least one well-documented example where the neighbour of a nucleotide may influence mutagenesis: CpG islands (i.e. dinucleotide sequences CG) have been mapped in mutation hotspots. Cytosines in such doublets may be targets for DNA methylation, producing 5-methyl-cytosine. 5-Methyl-cytosine may then undergo oxidative deamination resulting in the conversion of the initial cytosine into a thymine. Since thymine is a chemical moiety naturally occurring in DNA, this mutation may escape the mechanisms of DNA mismatch repair systems and has a high propensity to be fixed. So, a G residue following a C residue increases the likelihood that the C residue be mutated into a T residue. Hence there is mounting evidence that there is at least a deterministic component in the process of mutagenesis. Having shown the efficiency of our model in predicting DNA sequence mutations in a fully deterministic world, our next attempt would be to incorporate probabilistic components in our system. Such an approach might yield a tool able to wisely predict the likelihood of certain mutations. The usefulness of the ability to predict even the likelihood of mutations is tremendous. We might be able to use available data (Ghedin et al., 2005) in order to foresee possible mutations of viruses and other pathogens. Such information might allow us to prepare ourselves for future possible outbreaks.

Our model focuses on the neighbour-dependent mechanics of DNA sequence changes without taking into account the processes of natural selection. Therefore, at least in the present version, our model is not suitable for the analysis of DNA sequences of functional genes during phylogenesis. However, it might be very useful for the analysis of the evolution of DNA sequences that evade the pressure of natural selection, such as non coding sequences, hypervariable DNA sequences (such as the D-loop of mitochondrial DNA) and viral DNA or RNA under conditions of lack of competition (e.g. when different viruses replicate in the same cells or in the presence of helper viruses; Furió et al., 2005). Alternatively, our system might be used for the analysis of mutagenesis caused by DNA polymerases that do not possess efficient proofreading activity *in vitro*, such as Taq polymerase employed in polymerase chain reactions (PCR). Importantly, such reactions are used in DNA engineering when it is intended to introduce mutations. In this vein, it is intriguing to design experiments that may provide real data (DNA sequences) that could be used for the calibration of our simulator and the GA for the selection of the CA evolution rule.

The use of modelling DNA sequences as CAs and the implementation of GAs for the analysis of DNA sequence evolution should not be restricted to efforts towards understanding the mechanisms of mutagenesis. For example, we envision that the development of CA models may be useful for analysing the self-organizing properties of DNA and possibly indicate new directions in the field of artificial intelligence for bioinformatics.

References

Adamatzky, A., 1994. Identification of Cellular Automata. Taylor & Francis, London.

Archilla, J.F.R., Christiansen, P.L., Gaididei, Yu.B., 2002. Interplay of nonlinearity and geometry in a DNA-related, Klein-Gordon model with long-range dipole–dipole interaction. Phys. Rev. E (Stat. Nonlinear Soft Matter Phys.) 65, 016609/1-6.

Arndt, P.F., Burge, C.B., Hwa, T., 2001. DNA sequence evolution with neighbor-dependent mutation. http://www.arXiv.org/pdf/physics/0112029.

Baake, E., Baake, M., Wagner, H., 1997. Ising quantum chain is equivalent to a model of biological evolution. Phys. Rev. Lett. 78, 559–562.

Baake, E., Baake, M., Wagner, H., 1998. Quantum mechanics versus classical probability in biological evolution. Phys. Rev. E 57, 1191–1192.

Baxevanis, A.D., Ouellette, B.F., 1998. Bioinformatics, a Practical Guide to the Analysis of Genes and Proteins. Wiley-Interscience, New York.

Bernadres, A.T., dos Santos, R.M.Z., 1997. Immune network at the edge of chaos. J. Theor. Biol. 186, 173–187.

Bieberich, E., 2000. Probing quantum coherence in a biological system by means of DNA amplification. BioSystems 57, 109–124.

Chou, K.C., Zhang, C.T., 1992. Diagrammatization of codon usage in 339 HIV proteins and its biological implication. AIDS Res. Hum. Retroviruses 8, 1967–1976.

Chou, K.C., Wei, D.Q., Du, Q.S., Sirois, S., Zhong, W.Z., 2006. Review: progress in computational approach to drug development against SARS. Curr. Med. Chem. 13, 3263–3270.

Cios, K.J., Mamitsuka, H., Nagashima, T., Tadeusiewicz, R., 2005. Computational intelligence in solving bioinformatics problems. Artif. Intell. Med. 35, 1–8.

Das, R., Mitchel, M., Crutchfield, J.P., 1994. A genetic algorithm discovers particle-based computation in cellular automata. In: Proceedings of the Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, pp. 344–353.

Durbin, R., Eddy, S., Krogh, A., Mitchison, G., 2000. Biological sequence analysis. In: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge.

Fogel, D.B., 1995. Evolutionary Computation. IEEE Press, New York.

Furió, V., Moya, A., Sanjuán, R., 2005. The cost of replication fidelity in an RNA virus. Proc. Natl. Acad. Sci. U.S.A. 102, 10233–10237.

Gao, L., Ding, Y.S., Dai, H., Shao, S.H., Huang, Z.D., Chou, K.C., 2006. A novel fingerprint map for detecting SARS-CoV. J. Pharm. Biomed. Anal. 41, 246–250.

Gaylord, R.J., Nishidate, K., 1996. Modeling nature. In: Cellular Automata Simulations with Mathematica. Telos Springer-Verlag, Berlin.

Ghedin, E., et al., 2005. Large-scale sequencing of human influenza reveals the dynamic nature of viral genome evolution. Nature 437, 1162–1166.

Goldberg, D.A., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Golo, V.L., Volkov, Yu.S., 2002. Tautomeric Transitions in DNA. Los Alamos Preprint Archive. http://www.xxx.lanl.gov/abs/cond-mat/0110599.

Hjort, M., Stafstrom, S., 2001. Band resonant tunneling in DNA molecules. Phys. Rev. Lett. 87, 228101–228104.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Michigan.

Iovine, G., D'Ambrosio, D., Di Gregorio, S., 2005. Applying genetic algorithms for calibrating a hexagonal cellular automata model for the simulation of debris flows characterised by strong inertial effects. Geomorphology 66, 287–303.

Jones, M.E., Thomas, S.M., Clarke, K., 1999. The application of a linear algebra to the analysis of mutation rates. J. Theor. Biol. 199, 11–23.

Kansal, A.R., Torquato, S., Harsh, IV.G.R, Chiocca, E.A., Deisboeck, T.S., 2000. Simulated brain tumor growth dynamics using a three-dimensional cellular automaton. J. Theor. Biol. 203, 367–382.

Karafyllidis, I., 1997. A model for the prediction of oil slick movement and spreading using cellular automata. Environ. Int. 23, 839–850.

Karafyllidis, I., 1998. A model for the influence of the greenhouse effect on insect and microorganism geographical distribution and population dynamics. BioSystems 45, 1–10.

Karafyllidis, I., 2004. Design of a dedicated parallel processor for the predicition of forest fire spreading using cellular automata and genetic algorithms. Engineering Applications of Artificial Intelligence 17, 19–36.

Karafyllidis, I., Thanailakis, A., 1997. A model for predicting forest fire spreading using cellular automata. Ecological Modeling 99, 87–97.

Kirby, K.G., 2002. Biological adaptabilities and quantum entropies. BioSystems 64, 33–41.

Kryachko, E.S., 2002. The origin of spontaneous point mutations in DNA via Lowdin mechanism of proton tunnelling in DNA base pairs. Int. J. Quantum Chem. 90, 910–923.

McFadden, J., Al-Khalili, J., 1999. A quantum mechanical model of adaptive mutation. BioSystems 50, 203–211.

Mitchel, M., Crutchfield, J.P., Hraber, P.T., 1994. Evolving cellular automata to perform computations. Physica D 75, 361–391.

Ogryzko, V.V., 1997. A quantum-theoretical approach to the phenomenon of directed mutations in bacteria. BioSystems 43, 83–95.

Patel, A.A., Gawlinski, E.T., Lemieux, S.K., Gatenby, R.A., 2001. A cellular automaton model of early tumor growth and invasion: the effects of native tissue vascularity and increased anaerobic tumor metabolism. J. Theor. Biol. 213, 315–331.

Salzberg, C., Antony, A., Sayama, H., 2004. Evolutionary dynamics of cellular automata-based self-replicators in hostile environments. BioSystems 78, 119–134.

Schwefel, H.P., 2002. Deep insight from simple models of evolution. BioSystems 64, 189–198.

Sirakoulis, G.Ch., Karafyllidis, I., Thanailakis, A., 2000. A cellular automaton model for the effects of population movement and vaccination on epidemic propagation. Ecol. Model. 133, 209–223.

Sirakoulis, G.Ch., Karafyllidis, I., Mizas, Ch., Mardiris, V., Thanailakis, A., Tsalides, Ph., 2003. A cellular automaton model for the study of DNA sequence evolution. Comput. Biol. Med. 33, 439–453.

Sirakoulis, G.Ch., Karafyllidis, I., Sandaltzopoulos, R., Tsalides, Ph., Thanailakis, A., 2004. An algorithm for the study of DNA sequence evolution based on the genetic code. BioSystems 77, 11–23.

Ulam, S., 1974. Some ideas and prospects in biomathematics. Ann. Rev. Biol. 12, 277–292.

Venter, J.C., et al., 2001. The sequence of the human genome. Science 291, 1304–1351.

von Neumann, J., 1966. Theory of Self-Reproducing Automata. University of Illinois Press, Urbana.

Wang, M., Yao, J.S., Huang, Z.D., Xu, Z.J., Liu, G.P., Zhao, H.Y., Wang, X.Y., Yang, J., Zhu, Y.S., Chou, K.C., 2005. A new nucleotide-composition based fingerprint of SARS-CoV with visualization analysis. Med. Chem. 1, 39–47.

Welch, B., Jones, K., Hobbs, J., 2003. Practical Programming in Tcl and Tk, 4th ed. Prentice Hall, Englewood Cliffs, NJ.

Xiao, X., Shao, S., Ding, Y., Huang, Z., Chen, X., Chou, K.C., 2005a. Using cellular automata to generate image representation for biological sequences. Amino Acids 28, 29–35.

Xiao, X., Shao, S., Ding, Y., Huang, Z., Chen, X., Chou, K.C., 2005b. An application of gene comparative image for predicting the effect on replication ratio by HBV virus gene Missense mutation. J. Theor. Biol. 235, 555–565.

Xiao, X., Shao, S.H., Ding, Y.S., Huang, Z.D., Chou, K.C., 2006a. Using cellular automata images and pseudo amino acid composition to predict protein subcellular location. Amino Acids 30, 49–54.

Xiao, X., Shao, S.H., Chou, K.C., 2006b. A probability cellular automaton model for hepatitis B viral infections. Biochem. Biophys. Res. Commun. 342, 605–610.

Yang, Y., Billings, S.A., 2000. Neighborhood detection and rule selection from cellular automata patterns. IEEE Trans. Syst. Man Cybernet. A 30, 840–847.

Zhang, C.T., Chou, K.C., 1993. Graphic analysis of codon usage strategy in 1490 human proteins. J. Protein Chem. 12, 329–335.

Zhang, C.T., Chou, K.C., 1994. Analysis of codon usage in 1562 *E. coli* protein coding sequences. J. Mol. Biol. 238, 1–8.