Gene Expression Programming for Evolving Two-Dimensional Cellular Automata in a Distributed Environment

César Manuel Vargas Benítez, Wagner Weinert, and Heitor Silvério Lopes

Abstract. This paper presents a novel distributed bio-inspired approach that uses Gene Expression Programming (GEP) to evolve transition rules for two-dimensional Cellular Automata (2D-CA). The 2D-CA are simulated in parallel using a master-slave distributed environment. The fitness function of the GEP ultimately measures the ability of a given CA to create a suitable solution for a complex Bioinformatics problem. To validate the proposed approach, extensive experiments were done dealing with a computationally expensive problem, that is considered to be one of the most important open challenges in Bioinformatics. Results of simulations show that the proposed approach was effective for the problem. Future works will investigate other distributed approaches of this approach, such as those based on General-Purpose Graphics Processing Units (GPGPU) or hardware-based accelerators. Finally, we believe that the method proposed in this work can be useful for other computational problems.

Keywords: Bio-inspired Computing, Distributed Computing, Gene-Expression Programming, Cellular Automata, Self-organization, Emergent Behavior, Contact Maps, Protein Folding Problem.

1 Introduction

Stephen Wolfram proposed a "New kind of Science" that is based on general types of rules that can be embodied in simple computer programs for reproducing

César Manuel Vargas Benítez · Heitor Silvério Lopes
Bioinformatics Laboratory, Federal University of Technology - Paraná (UTFPR), Curitiba, Brazil
e-mail: {cesarbenitez,hslopes}@utfpr.edu.br

Wagner Weinert
Federal Institute of Education Science and Technology of Paraná (IFPR), Paranaguá, Brazil
e-mail: wrweinert@gmail.com

© Springer International Publishing Switzerland 2015
D. Camacho et al. (eds.), *Intelligent Distributed Computing VIII*,
Studies in Computational Intelligence 570, DOI: 10.1007/978-3-319-10422-5_12

real-world complex behaviors, instead using traditional mathematical methods [17]. A particular class of such computer program are the Cellular Automata (CAs), which are simple discrete idealizations of natural systems. CAs are families of simple, finite-state machines that exhibit emergent behaviors through their interactions [8].

The computational simulation of a CA system is relatively simple, where a configurational state of the CA is determined according to its predecessor state and a transition rule. However, finding a transition rule for a given dynamic behavior is a very difficult task, for which there is no efficient method [16].

The main objective of this work is to propose a novel parallel approach for the induction of transition rules of two-dimensional Cellular Automata (2D-CA), using Gene Expression Programming, for solving complex problems in a reasonable computing time. The second objetive, not less important, is to apply the proposed approach to the Protein Contact Map prediction, validating the proposed approach and proposing a novel method for the Protein Folding Prediction Problem (PFP).

This paper is organized as follows: Section 2 presents an overview about Cellular Automata; Section 3 describes the Gene Expression Programming (GEP); Section 4 presents an overview about the PFP; Section 5 describes the proposed approach; Section 6 shows how the experiments were conducted and the results; finally, in Section 6 conclusions and future works are presented.

2 Cellular Automata (CA)

Cellular Automata (CAs) were introduced by John von Neumann in his work on *self-reproducing machines* [12] and have been used to model several biological, physical and engineering systems. For instance, simulation of the HIV infection dynamics [19], and water flow simulation [14]. Basically, CAs are discrete dynamic systems that are be represented by a d-dimensional array, composed of identical interconnected components (cells).

The dynamic behavior of a CA is represented by its spatio-temporal diagram [17]. Each cell has a discrete state that is updated on discrete time steps, considering its current state and the state of the neighboring cells (neighborhood relationship). All cells of the d-dimensional matrix are updated at the same time step by the application of a transition rule. Thus, sucessive applications of the transition rule will lead to a dynamic behavior, from the initial state in t_0 to successive states in subsequent time steps $(t_1, \cdots, t_n)$.

The following formal notation for CAs is presented by [11]: $\sum$ set of possible states of a cell; k number of elements of the set $\sum$; i index of a specific cell; S_i^t state of a cell in a given time t; η_i^t neighborhood of cell i; $\Phi(\eta_i^t)$] transition rule that defines the next state S_i^{t+1} for each cell i, as function of η_i^t.

The neighborhood of each cell $(c_{i,j})$ of a two-dimensional Cellular Automaton (2D-CA) is composed following a neighborhood relationship. The neighborhood relationship is determined by a predefined radius (r) and the size (number of cells) of the neighborhood (m) is defined as a function of r, acoording to $m = 2r + 1$.

The most common types of neighborhood for 2D-CA are the von Neumann and the Moore neighborhoods. The size of the von Neumann neighborhood with $r = 1$ is $m = 5$, comprising the four orthogonally neighboring cells ($c_{i-1,j}$, $c_{i,j-1}$, $c_{i+1,j}$, $c_{i,j+1}$) surrounding a central cell ($c_{i,j}$). On the other hand, the Moore neighborhood is composed by the central cell and eight neighboring cells ($c_{i-1,j}$, $c_{i,j-1}$, $c_{i+1,j}$, $c_{i,j+1}$, $c_{i-1,j-1}$, $c_{i+1,j-1}$, $c_{i-1,j+1}$, $c_{i+1,j+1}$).

In addition, boundary conditions are used to allow the connection between cells that are situated at the extremities, forming a toroidal arrangement. Thus, the transition rule ($\Phi(\eta_i^t)$) is applied over all cells of the CA, without failure.

The number of transitions (possible configurations of the neighborhood) that compose the rule Φ is given by k^{2r+1} and the number of rules represented by those transitions is $k^{k^{2r+1}}$. For instance, the rule of a binary 2D-CA with von Neumann neighborhood (with $r = 1$) is composed of 32 transitions. In other words, the rule is composed of 32 bits, where each bit represents the result of a transition (i.e. $c_{i,j}^{(t)} \rightarrow c_{i,j}^{(t+1)}$), according to the concept of elementary automata proposed by Wolfram.

3 Gene-Expression Programming (GEP)

Gene-Expression Programming (GEP) is an extension of Genetic Programming (GP) that was proposed by [5]. The difference between these approaches lies in how the individuals are represented. In GP, the individuals are nonlinear entities of different sizes and shapes (concrete syntax trees). On the other hand, in GEP the individuals are encoded as fixed-size linear strings (also known as genome or chromosome), which are afterwards expressed as nonlinear entities of different sizes and shapes (i.e. expression trees or diagram representations) [5].

The encoding of individuals in GEP is based on the biological concept of open read frame (ORF), that is the coding sequence of a gene. However, it is important to know that genes are composed of more sequences than the respective ORF. In biology, an ORF is composed of amino acid codons, beginning with a "start" codon and ending at a termination codon. GEP genes are composed of a head and a tail. The head has symbols that represent functions and terminals. On the other hand, the tail contains only terminals. GEP genes can have noncoding regions, that, in fact, are the essence of GEP, allowing modifications of the genome using any genetic operator without restrictions, producing valid programs without the need for editing processes [5]. In other words, the encoding region of a gene (ORF) can "activate" or "deactivate" portions of the genetic material algorithm, according to the functions and their arities (i.e. number of arguments) encoded in the head of the gene.

GEPCLASS [18] is an implementation of GEP specially designed for finding rules for classification problems based on supervised learning, where it is aimed to find rules for modeling a given domain of known data samples, and then classify unseen sets of data. In GEPCLASS, a population of individuals evolves for a number of generations, where selected individuals are subjected to genetic operators (mutation, recombination and transposition), generating diversity and, consequently, allowing the evolutionary process to continue for more generations, increasing the chances of finding even better solutions. The head of the gene can have elements belonging only to the set of functions, such as logical and comparison operators.The tail, in turn, can have elements either from the set of functions or from the set of terminals, which in turn, includes the attributes that describe particular values.

The mapping between the genotype to the phenotype is carried out as follows. The chromosome is transcribed into a variable-size expression tree (ET), following the Karva language [5], where each gene is trascribed to a separated sub-tree. Then, all sub-trees are joined together by a linking function (*AND* or *OR* operator), composing the ET that represents a candidate solution to a given problem. The quality of the candidate solutions is measure by a fitness function.

4 The Protein Folding Problem and the Protein Contact Maps

Under physiological conditions every protein folds into a unique three-dimensional structure, also known as the native tertiary structure or native conformation, that determines their specific biological function. This process is known as the Protein Folding.

Despite the considerable theoretical and experimental effort expended to study the protein folding process, there is not yet a detailed description of the mechanisms that govern the folding process.

Although the concept of the folding process arose in the field of Molecular Biology, the Protein Folding Problem (PFP) is clearly interdisciplinary, requiring support of many knowledge areas, and it is considered to be one of the most important open challenges in Biology and Bioinformatics.

Better understanding the protein folding process could help to: (a) accelerate drug discovery by replacing slow, expensive structural biology experiments with faster computational simulations, and (b) infer protein function from genome sequences.

Contact Maps (CM) are minimalistic representations of protein structures. The contact map for a protein sequence with N amino acids is a $N \times N$ binary symmetrical matrix (C), which is defined as follows: $C_{i,j} = 1$ if residues ith and jth are in contact, otherwise $C_{i,j} = 0$. Each position of the matrix (ith,jth) is 1 if the amino acid pair (ith and jth amino acids) fulfills the connectivity condition. One can define a contact between two amino acids in different ways. For instance, we can consider two amino acids in contact when their Cα atoms are closer than a arbitrary threshold distance [4].

As commented in last section, the solution of the folding problem is still lacking. Among different possibilities, the prediction of protein contact maps is particularly promising, since even a partial solution of it can significantly help the prediction of the protein structure [4]. Several methods have been developed for CM prediction from sequence. For instance, Neural Networks (NN) [7], Genetic Programming (GP) [9] and neuro-fuzzy systems [1].

5 Implementation of the Parallel GEP-CA (pGEP-CA)

Algorithm 2 shows the pseudo-code of the pGEP-CA. A parallel master-slave architecture is employed in order to allow a reasonable computing time. The processing load is divided into several processors (slaves), under the coordination of a master processor. The master is responsible for initializing the population, determining the ORFs, performing the selection procedure, applying the operators(clone operator, mutation, recombination, IS (insertion sequence) transposition, RIS (root IS) transposition and genic transposition) based on the GEPCLASS [18] and distributing individuals to the slaves. Slaves, in turn, are responsible for reading the initial and final 2D-CAs, simulating the 2D-CAs from the initial 2D-CA, using the induced rules and computing the fitness function of each individual received, using the final (expected) 2D-CAs and the obtained 2D-CAs. In Algorithm 2, bold instructions are processed in parallel. The software was developed in ANSI-C programming language, using the Message Passing Interface (MPI) for communication between processes[1] and the Mersenne Twister random number generator [10].

Algorithm 2. Pseudo-code for parallel GEP-CA (pGEP-CA)

```
 1: Start
 2: Initialize population;
 3: Determine ORFs
 4: Simulate 2D-CAs and Evaluate fitness in parallel
 5: while stop criteria not satisfied do
 6:     Clone operator – part 1
 7:     Selection
 8:     Apply genetic operators
 9:     Update population
10:     Determine ORFs
11:     Simulate 2D-CAs and Evaluate fitness in parallel
12:     Clone operator – part 2
13: end while
14: Export postprocess results: best transition rule, obtained CM, metrics
15: End
```

5.1 Solution Encoding and Fitness Function

First of all, it is important to know how the transition rule is composed. As commented in Section 2, the transition rule is formed by concatenating all transitions, which in turn, are defined by the possible combinations of the neighboring cells. The number of combinations, using the von Neumann neighborhood with unity radius, is 32, obtained as shown in Section 2. For instance, the rule

[1] Available at: `http://www.mcs.anl.gov/research/projects/mpich2/`

"$01010111011111111011111111011001012_2$" is formed by the transitions, from right to left, "00000_2"$\rightarrow 1$; "00001_2"$\rightarrow 0$; "00010_2"$\rightarrow 1$; $\cdots$; "11111_2"$\rightarrow 0$.

In this work, the encoding of the individuals is defined according to the set of terminals and their domains, following the Pittsburgh approach [6]. The terminals are binary and represent the state of the neighboring cells ('1'=contact, '0'=non-contact). The set of terminals represent all possible combinations of the neighborhood, mapping all transitions of a given rule. Considering the von Neumann neighborhood with $r = 1$, the terminal set is composed of five terminals (labeled as a, b, c, d and e). The terminals have the same domain, which in turn, is defined by the possible states of the cells of the CA ($\sum = [0; 1]$).

The individuals are represented by two multigenic chromosomes, which in turn, are composed of more than one gene of equal size. As stated in Section 3, every gene is divided into a head and a tail. The size of the head (h) of each gene and the number of genes of each chromosome can be chosen *a priori*. On the other hand, the size of the tail (t) is determined according to the size of the head as proposed by [18]: $t =$ IntegerPart$[0.5(h(n-1)+1)]$, where n represents the largest arity (number of arguments) of the functions used.

Each gene is directly translated into an expression tree (ET). In this work, each chromosome is composed of two genes. Thus, the sub-ETs codified by the genes are linked together by a logical function (*AND* or *OR*), which can be chosen *a priori*.

A possible transition encoded in an individual is written in the form *IF A THEN C* as in data classification systems. For example, a possible rule encoded in an individual would be: *IF (*$a = 1$ *AND* $b = 0$*) THEN Rule = '1'; else Rule = '0'*.

Figure 1 shows a simplified example of the transcription process.

Contact maps (CMs) are generally sparse symmetric matrices, populated primarily with non-contacts (or zeros). Therefore, a similarity measure between two CMs based on the Hamming distance [13] or Euclidean distance does not work well for CMs, because contacts (true) and non-contacts (false) values carry the same weight. Therefore, the fitness function proposed in this work is based on three metrics, that are better suited to this problem, chosen to measure the ability of a transition rule to generate a CA that represents a CM correctly. The fitness function is shown in Equation 1.

$$fitness = S_C * S_{NC} * S_i^2 \tag{1}$$

where: S_C, S_{NC} are based on the sensitivity and specificity measures, respectively. Sensitivity and specificity are commonly used in classification systems. Sensitivity measures the ability of the classifier to correctly assign a data to its real class. On the other hand, specificity measures the ability to reject a given data as belonging to a class to which it does not belong. In this work, S_C and S_{NC} measure the ability of a transition rule to generate correct contacts and non-contacts, respectively. S_C and S_{NC} are defined following four types of result, as shown in Equations 2 and 3, respectively. S_i measures the symmetry of the CM, as shown in Equation 4, where m and n are the number of rows and columns of the CM, respectively.

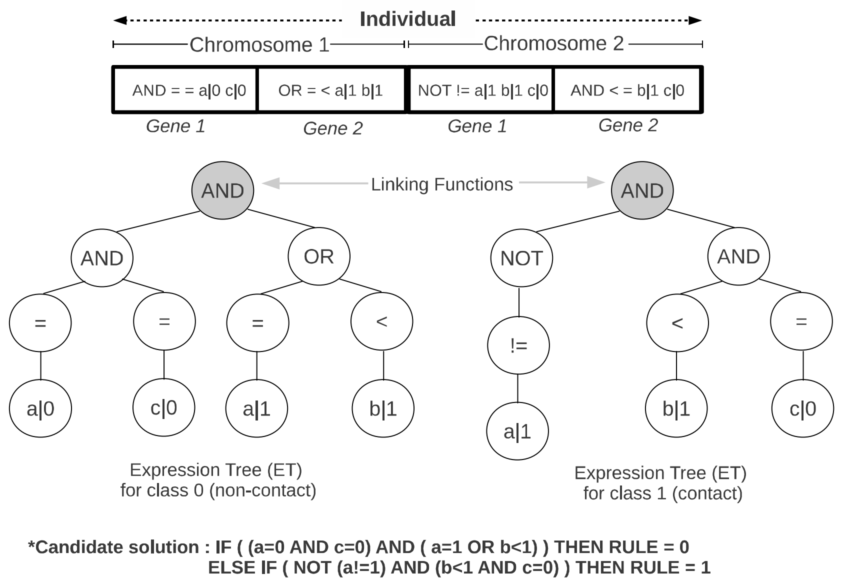

Individual
Chromosome 1
Chromosome 2
AND = = a|0 c|0
OR = < a|1 b|1
NOT != a|1 b|1 c|0
AND < = b|1 c|0
Gene 1
Gene 2
Gene 1
Gene 2
Linking Functions
AND
AND
OR
=
=
=
<
a|0
c|0
a|1
b|1
AND
NOT
AND
!=
<
=
a|1
b|1
c|0
Expression Tree (ET) for class 0 (non-contact)
Expression Tree (ET) for class 1 (contact)
*Candidate solution : IF ((a=0 AND c=0) AND (a=1 OR b<1)) THEN RULE = 0
ELSE IF (NOT (a!=1) AND (b<1 AND c=0)) THEN RULE = 1

Fig. 1 GEP Transcription process – example

$$S_C = \frac{T_C}{T_C + F_{NC}} \quad (2) \qquad S_{NC} = \frac{T_{NC}}{F_C + T_{NC}} \quad (3) \qquad S_i = \sum_{i=0}^{i<m-1} \sum_{j=i+1}^{j<n} [1 - |(C_{i,j} - C_{j,i})|] \quad (4)$$

where:

- **True contacts (T_C):** number of contacts generated by the transition rule that, in fact, are contacts;
- **True non-contacts (T_{NC}):** number of non-contacts generated by the transition rule that, in fact, are non-contacts;
- **False contacts (F_C):** it counts the contacts generated by the transition rule that, in fact, are non-contacts;
- **False non-contacts (F_{NC}):** it counts the non-contacts generated by the transition rule that, in fact, are contacts;
- $C_{i,j}$ represents the value at cell location (i, j) of the CM.

6 Computational Experiments and Results

All experiments done in this work were run in a cluster of networked computers. Each computer has an Intel Core-2 Quad processor at 2.8GHz. All computers run a minimal installation of Arch Linux [2].

The contact maps (CMs) used in the experiments to validate our proposed approach – the pGEP-CA – were generated by Molecular Dynamics (MD) simulations following our previous work [2], using the 3DAB model of the protein 2gb1, which is composed by 56 amino acids. Each CM is a 56x56 matrix and represents a folding state of the folding process. In this work, 100 CMs were generated for the following threshold values: 6.65, 7, 8, 9, 10, 11 and 12Å. Thus, a total of 700 CMs were generated.

In this work, CMs are represented by 2D-CAs, which in turn, are simulated using evolved (or induced) transition rules by GEP simulations. In order to evaluate the proposed approach, experiments were done using consecutive ith and jth CMs, which in turn, represent the initial 2D-CA configuration and the expected final 2D-CA configuration, respectively. The fitness of the GEP individuals is computed using the expected CM (obtained by MD simulations) and the achieved 2D-CA built using the obtained rules. Due to the stochastic nature of the algorithm, 30 independent runs were done with different initial random seeds. Thus, a total of 20790 experiments were done.

The running parameters for the pGEP-CA are: population size ($Pop = 100$), number of GEP generations ($Maxgen = 350$), linking function ($link$ =AND), function set = [AND, OR, NOT], terminal set = [a, b, c, d, e], number of genes per chromosome ($n_g = 2$), size of head ($h = 10$), selection method sel, genetic operators probabilities ($pcross = 0.8$ – recombination, $pmut = 0.1$ – mutation, $ptIS = 0.7$ – IS transposition, $ptRIS = 0.7$ – RIS transposition, $pgt = 0.7$ – genic transposition), number of 2D-CA interactions ($CAiter = 1$), 2D-CA von Neumann neighboorhood (with $r = 1$, $m = 5$) and number of slaves ($s = 50$).

The results of our experiments are shown in Table 1. Some results are not shown here due to space restrictions. In this table, the first column shows the metrics. Next columns show their values for each threshold value. We can observe, from the $fitness$, S_C and S_{N_C} values, that the induced transition are able to generate 2D-CAs with correct contacts and non-contacts, despite the F_C and F_{NC}. It can also be observed that parallel processing was essential for obtaining results in reasonable processing time (t_p).

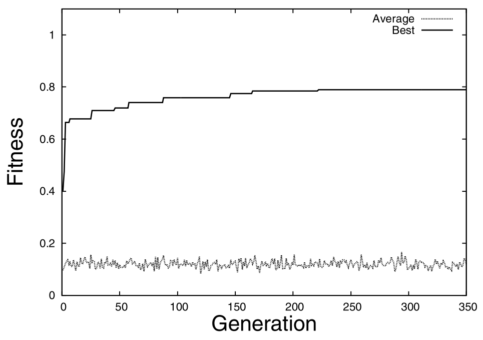
Figure 2(a) shows a plot of the fitness (best and average) obtained in a simulation. In this figure, that the genetic diversity is preserved, since the distance from average to best is maintained along GEP generations. Results can be improved hybridizing the pGEP-CA with specialized strategies in order to keep high genetic diversity and explore the search space efficiently leading to better individuals, such as local search methods or coevolution with other Evolutionary Computation algorithms as proposed by [3]. Figures 2(b) and (c) show a expected (final) CM and the obtained

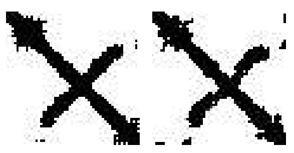
[2] Available in: `https://www.archlinux.org`

CM generated using the induced rule ("$01111111011111110111111101111111_2$", with $fitness = 0.85$). The obtained CM suggest that the proposed fitness function is adequate to induce transition rules for evolving 2D-CAs, which in turn, represent CMs. We believe that best results can be obtained, using a knowledge-based strategy to correct the faults of the obtained CMs.

Table 1 Numerical results obtained using CMs with different threshold values

Metric	CM threshold [Å]			
$Avg(Min/Max)$	6.65	9	10	12
Best $fitness$	0.91 (0.73/0.97)	0.92 (0.76/0.97)	0.897 (0.76/0.93)	0.89 (0.74/0.95)
T_C	393.3 (216/428)	1323.4 (556/1392)	1548.6 (650/1644)	2160.4 (770/2280)
F_C	34.1 (12/277)	63.1 (12/211)	78.62 (20/202)	61.12 (2/182)
T_{NC}	2677.2 (2493/2789)	1682.6 (1608/2408)	1424.4 (1328/2307)	834.9 (730/2130)
F_{NC}	31.4 (12/277)	66.9 (28/206)	84.4 (32/218)	79.64 (26/338)
S_C	0.92 (0.76/0.98)	0.95 (0.76/0.98)	0.95 (0.78/0.98)	0.96 (0.74/0.988)
S_{NC}	0.98 (0.9/0.996)	0.96 (0.9/0.99)	0.95 (0.90/0.99)	0.93 (0.87/0.999)
S_i	0.9998	0.9999	0.99997	0.99999
Avg t_p(s)	12.68	12.69	12.70	12.69





(a) (b)

Fig. 2 (a) Example of performance of the pGEP-CA and Example of an obtained CM: (b) Expected CM (c) Obtained CM, where cells in states '0' and '1' are represented by white and black squares (or dots), respectively.

7 Conclusions

The process of 2D-CA transition rules induction for simulating dynamic behaviors is still an open research problem. Therefore, the approach presented in this work represents an important contribution regarding this issue.

As commented in last section, the hybridization with specialized strategies, such as local search methods and coevolution with other EC algorithms will be focused in future works.

Regarding the processing time for the simulations, future research will need highly parallel approaches for dealing with the problem, such as the use of GPGPU (General Purpose Graphics Processing Units) [15].

This work also contributes significantly to Bioinformatics, presenting the first implementation of a parallel computational approach based on GEP and CA applied to the Contact Map Prediction.

In a broader sense, we believe that the proposed approach presented in this paper is very promising for the research areas related to Cellular Automata and Protein Folding Problem.

References

1. Abu-Doleh, A., Al-Jarrah, O., Alkhateeb, A.: Protein contact map prediction using multi-stage hybrid intelligence inference systems. Journal of Biomedical Informatics 45, 173–183 (2012)
2. Benítez, C.M.V., Lopes, H.S.: Ab-initio protein folding using molecular dynamics and a simplified off-lattice model. Journal of Bionanoscience 7, 391–402 (2013)
3. Benítez, C.M.V., Parpinelli, R., Lopes, H.: A heterogeneous parallel ecologically-inspired approach applied to the 3D-AB off-lattice protein structure prediction problem. In: BRICS Countries Congress (BRICS-CCI) and Brazilian Congress (CBIC) on Computational Intelligence (2013)
4. Fariselli, P., Olmea, O., Valencia, A., Casadio, R.: Progress in predicting inter-residue contacts of proteins with neural networks and correlated mutations. Proteins 45, 157–162 (2001)
5. Ferreira, C.: Gene expression programming: a new adaptive algorithm for solving problems. Complex Systems 13, 87–129 (2001)
6. Freitas, A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer (2002)
7. Lena, P.D., Nagata, K., Baldi, P.: Deep architectures for protein contact map prediction. Bioinformatics 28(19), 2449–2457 (2012)
8. Luger, G.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 6th edn. Addison-Wesley Publishing Company, USA (2008)
9. MacCallum, R.: Striped sheets and protein contact prediction. Bioinformatics 20(1), I224–I231 (2004)
10. Matsumoto, M., Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation 8(1), 3–30 (1998)
11. Mitchell, M.: Computation in cellular automata: a select review. In: Nonstandard Computation, 1st edn., pp. 95–140. VHC Verlagsgesellschaft, Weinheim (1998)
12. Neumann, J.V.: Theory of self-reproducing automata. University of Illinois Press (1966)
13. Peterson, W., Weldon, E.: Error-correcting codes. MIT Press (1972)

14. Topa, P., Mlocek, P.: Using shared memory as a cache in cellular automata water flow simulations on gpus. Computer Science 14, 385–401 (2013)
15. Scalabrin, M., Parpinelli, R., Benítez, C.M.V., Lopes, H.: Population-based harmony search using GPU applied to protein structure prediction. International Journal of Computational Science and Engineering 9, 106–118 (2014)
16. Weinert, W., Lopes, H.S.: Evaluation of dynamic behavior forecasting parameters in the process of transition rule induction of unidimensional cellular automata. BioSystems 99, 6–16 (2010)
17. Wolfram, S.: A New kind of science. Wolfram Media, Champaign (2002)
18. Weinert, W.R., Lopes, H.S.: GEPCLASS: A classification rule discovery tool using gene expression programming. In: Li, X., Zaïane, O.R., Li, Z.-h. (eds.) ADMA 2006. LNCS (LNAI), vol. 4093, pp. 871–880. Springer, Heidelberg (2006)
19. Mo, Y., Ren, B., Yang, W., Shuai, J.: The 2-dimensional cellular automata for HIV infection. Physica A 399, 31–39 (2014)