
Author(s)

S K Tan and S U Guan

Title

Evolving Cellular Automata to Generate Nonlinear Sequences with Desirable Properties

Original Citation

Syn Kiat Tan and Sheng-Uei Guan, “Evolving Cellular Automata to Generate Nonlinear Sequences with Desirable Properties”, Applied Soft Computing, Volume 7, Issue 3, June

Evolving Cellular Automata to Generate Nonlinear Sequences with Desirable Properties

Syn Kiat Tan[1] and Sheng-Uei Guan[2]

[1]Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent, Singapore 119260

[2]School of Engineering and Design
Brunel University
Uxbridge, Middlesex, UB8 3PH, UK

Abstract—This paper presents a new chromosomal representation and associated genetic operators for the evolution of highly nonlinear cellular automata that generate pseudorandom number sequences with desirable properties ensured. This chromosomal representation reduces the computational complexity of genetic operators to evolve valid solutions while facilitating fitness evaluation based on the DIEHARD statistical tests.

Index Terms— cellular automata, random number generation, incremental evolution.

I. INTRODUCTION

The theory for cellular automata (CA) based pseudorandom number generators (PRNG) is well developed [1,2] and n-bit linear CA can be designed to generate sequences with desirable properties: maximum period $p = 2^n - 1$, uniform distribution of n-bit tuples and balanced distribution of '1' and '0'. These so-called m-sequences cannot be used directly due to their linear structure while the theory for nonlinear CA are less developed since it

is difficult to ensure desirable sequence properties [8]. For cryptographic applications, a nonlinear Boolean function must be applied to destroy the linearity within these m-sequences. The Boolean functions used must possess high nonlinearity, high algebraic degree, correlation-immunity etc [8]. Cryptographic Boolean functions are usually very complex and large. Recently, heuristic methods have been gaining popularity in the search for such Boolean functions [6]. In this paper, we evolve nonlinear CA by viewing the linear CA and nonlinear Boolean function as a single entity.

In another direction, researchers focused on a variety of genetic algorithm (GA) based techniques to improve the randomness quality of CA generated sequences – typically verified though empirical testing with the DIEHARD statistical test suite [7]. The cellular programming approach [3] is used to evolve the linear transformation function of individual registers in the CA so that the generated sequences pass all 19 DIEHARD test results. In [4], the above approach is extended to consider nonlinear functions with up to five inputs. In [5], multi-objective genetic algorithm is used for evolving the minimum cost CA to pass all 19 DIEHARD tests. Due to complex CA models evolved, it is generally difficult to apply analysis to properties such as period length for these CA designs.

Approaches combining GA and CA have shown tremendous potential in various disciplines [12,13]. In our context, CA based PRNG that pass DIEHARD have been successfully evolved via GA [3-5]. However, such GA derived solutions currently lack

the theoretical support possessed by the m-sequences generated by linear CA. Our objectives thus follow:

i) to provide an efficient chromosomal representation and genetic operators that ensure evolved solutions have desirable sequence properties: long period, balanced distribution of '1' and '0', uniform distribution of *n*-bit output,

ii) to evolve solutions that pass all 19 DIEHARD tests.

II. CELLULAR AUTOMATA

An *n*-bit CA is an array of *n* binary registers whose state at time (t) can be denoted by $S^{(t)} = [s_{n-1}^{(t)}, s_{n-2}^{(t)}, \ldots, s_0^{(t)}]'$ (see Fig. 1). Each CA register s_i has a transformation function $f_i(.)$ to compute its next state, i.e. $f_2 = s_3^{(t)} \oplus s_1^{(t)}$ in Fig. 1. The CA can be equivalently viewed as a vector Boolean function $F \equiv [f_{n-1}(.), f_{n-2}(.), \cdots, f_0(.)]'$ which maps the current CA state to the next state, i.e. $S^{(t+1)} = F(S^{(t)})$. In Fig. 1, if $S^{(t)} = [1,1,1,1]'$, we have $S^{(t+1)} = [0,0,1,0]'$. The states of a CA during each discrete time step can be successively sampled to form a pseudorandom *n*-bit word sequence $\{S^{(0)}, S^{(1)}, S^{(2)}, \ldots\}$.

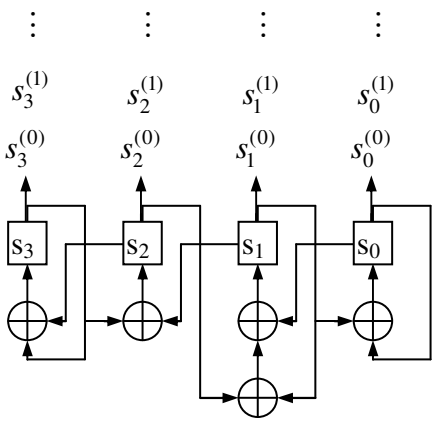

Fig. 1. A 4-bit linear CA using only XOR gates

A symmetric neighborhood is often used to denote the surrounding CA registers that can be used as inputs for each register. For example, with neighborhood radius *r=1*, the CA in Fig. 1 has $s_i^{(t)} = f_i(s_{i-1}^{(t)}, s_i^{(t)}, s_{i+1}^{(t)})$, $0 \le i \le n-1$. We only consider CA with null boundary conditions where the leftmost/rightmost registers' function receive a fixed "0" input from its "supposed" left/right neighbors respectively, i.e. $s_0^{(t)} = f_0(s_0^{(t)}, s_{i+1}^{(t)})$ and $s_{n-1}^{(t)} = f_{n-1}(s_{n-2}^{(t)}, s_{n-1}^{(t)})$. Null boundary conditions avoid long connection wires routing across the whole length of the CA when periodic boundary conditions are used

Denote $S^{(t)}$ as the *n*-bit output from the CA at time *(t)*. Consider the sequence $\{S^{(1)}, S^{(2)}, \ldots, S^{(2^n-1)}\}$ where each unique *n*-bit $S^{(t)}$ appears exactly once. It is clear that there are $(2^n-1)!$ such sequences that can be generated - these include both the linear m-sequences generated by CA as well as nonlinear sequences (generated by a system with nonlinear functions). Ideally, these sequences should form the complete search space for genetic algorithms to locate solutions that pass all DIEHARD tests. However, in previous approaches [3-5], solutions are not searched specifically within this solution space due to the chromosomal representation used.

In [3-5], the chromosomal representation used for evolving CA-based PRNG [3-5] is the concatenated binary string containing the truth tables of each CA registers' transformation functions $F \equiv [f_{n-1}(.), f_{n-2}(.), \cdots, f_0(.)]'$. These individual functions $f_i(.)$ are constrained to use *r=1* neighborhood. This means only $2^{2^{r+1}}$ functions can be

represented by this truth table. This reduces the size of the chromosome (concatenated truth tables) to $n \cdot 2^{r+1}$ bits and the evolutionary search space is focused on CA candidates with simple local inter-connections. There are also some works that extends this neighboring radius *r* to 5 so that a larger search space can be exploited [4]. Nevertheless, it is possible to evolve CA that passes all DIEHARD tests as shown in prior work, but if desirable sequence properties are required, the present chromosomal representation is not appropriate. Information about the sequence properties is not readily present in the truth table. In the following section, we introduce a more appropriate chromosomal representation that avoids the decision for the constraint *r*.

III. THE GENETIC ALGORITHM

A variety of heuristic methods, including genetic algorithm, simulated annealing, hill climbing etc., has been examined [6] to search for Boolean functions satisfying several desirable cryptographic properties. The evolved functions located possess properties which compares favorably with functions built from analytical approaches [10].

A. Chromosomal Representation

All *n*-input Boolean functions can be represented by two commonly used chromosomal representations: truth table representation (TTR) and algebraic normal form representation (ANFR) [9]. The TTR is simply the column vector for the truth table output over all possible inputs in lexicographical order. The ANFR is defined as the vector of coefficients from the algebraic normal form (XOR sum of AND terms) of the

Boolean function [8]. Both representations use 2^n bits for an n-input Boolean function and an efficient transform between these two representations is given in [9].

The TTR is popular with researchers on evolution design of Boolean functions [9] and CA [3-5] as it is intuitive and direct. The balanced-ness [8] of the Boolean function can also be easily computed from the TTR. The ANFR is suitable for studying the structure of the Boolean function in terms of the XOR and AND logic involved. The algebraic degree can be easily read from the ANFR.

There is no clear way to determine the period for generated sequences from both TTR and ANFR. Furthermore, it is difficult to apply the common crossover and mutation operators on both the TTR and ANFR without losing any of the desired sequence properties. Each newly generated child chromosome has to be checked and discarded if violations occur. This is clearly very inefficient. On the other hand, it is also unclear how to design a new genetic operator that will work with the TTR and ANFR that will ensure child chromosomes automatically possess the desired sequence properties.

The maximum length property implies the overall set of n Boolean functions $F \equiv [f_{n-1}(.), f_{n-2}(.), \cdots, f_0(.)]'$ must be considered simultaneously but both TTR and ANFR work only on a single Boolean function $f_i(.)$. Since we want all possible n-bit tuples in a single period, the proposed transition representation (TR) is simply the generated sequence itself, denoted as $[S^{(1)}, S^{(2)}, ..., S^{(2^n-1)}]$. Each gene $S^{(t)}$ is the decimal equivalent of the corresponding CA output at time *(t)*. As long as each

$S^{(t)} \in \{1, 2, ..., 2^n - 1\}$ appears exactly once in the chromosomal representation, the following desirable properties are ensured: maximum period $p = 2^n - 1$, uniform distribution of *n*-bit tuples and balanced distribution of '1' and '0'. (The actual CA implementation that generates the sequence can be obtained by a transform from the TR to the ANFR [9].)

The TR allows efficient initialization of the starting population of chromosomes. We first create a chromosome using a known linear CA that generates an m-sequence (this can be a simple counter that starts from 1 to $2^n - 1$). This guarantees a valid chromosome and genetic operations can be applied on this chromosome to further breed the entire starting population of valid chromosomes. For efficiency in ensuring child chromosomes are valid, all genetic operations must work with the natural boundaries set by the *n*-bit genes. In other words, each *n*-bit gene is the smallest unit that can be manipulated by the genetic operators. The following replacement-swapping operator ensures that all evolved sequences retain the desirable sequence properties.

B. The New Genetic Operator - Replacement-Swapping

The popular crossover operator is used to produce child chromosomes using blocks of genes from two parents. Crossover is essentially a global search operator and its usefulness is widely debated [11] and can be very problem-specific. Using ordinary crossover directly with TR will (with very high probability) result in an invalid child chromosome. To create a child that is very similar to two parents, a new operator replacement-swapping is suggested in Fig. 2. The child is a near-duplicate of the higher-

fitness parent-A except the following: ρ consecutive genes are replaced using the weaker parent-B's genes and ρ genes distributed within the parent-A are also swapped at the same time. These coordinated replacement and swapping operations ensures that all desirable sequence properties are retained.

Initialize child = parent-A (parent-A's fitness higher than parent-B)
Randomly select a starting point $i \in [0, 2^n - 1]$ and the range ρ of genes
For each $j = i + \rho \bmod 2^n$,
    Child $S^{(j)}$ = parent-B $S^{(j)}$ // replace operation
    Child $S^{(k)}$ = parent-A $S^{(j)}$ // swap operation, (k is the prior position of Child $S^{(j)}$)
End

Fig. 2. The replacement-swapping operator

Fig. 3 shows an example of how the replacement-swapping operator is used to achieve the "crossover" effect. Based on a 4-bit CA, each $S^{(t)} \in \{1, 2, ..., 15\}$ and we use a random starting point $i=3$ with $\rho = 5$. Genes in bold represent changes made to the child chromosome.

parent_A	3	15	1	7	14	5	6	13	8	9	11	10	2	4	12	
parent_B	1	7	3	4	12	6	2	8	9	15	11	10	13	14	8	
child	3	15	1	7	14	5	6	13	8	9	11	10	2	4	12	Step 1
child	**1**	15	**3**	7	14	5	6	13	8	9	11	10	2	4	12	Step 2
child	**1**	15	**3**	**4**	14	5	6	13	8	9	11	10	2	**7**	12	Step 3
child	**1**	15	**3**	**4**	**12**	5	6	13	8	9	11	10	2	**7**	**14**	Step 4
child	**1**	15	**3**	**4**	**12**	**6**	**2**	13	8	9	11	10	2	**7**	**14**	Step 5
child	**1**	15	**3**	**4**	**12**	**6**	**2**	13	8	9	11	10	2	**7**	**14**	Step 6

Fig. 3. Replacement-swapping for chromosomes of a 4-bit CA

Without a local search mechanism, a GA search is likened to the randomized search and an optimum can be difficult to locate. The mutation operator performs local search around the best performing solutions during evolution by "flipping" bits in the chromosome according to the mutation probability p_m. Here, any changes must be made in pairs such that all n-bit states remain distinct in order to avoid violating the maximum length property. The easiest way is to use still the replacement-swapping operator. Crossover's global search effect is achieved by using a block version of replacement-swapping while mutation's local search effect is achieved by using a point version of replacement-swapping, i.e. use $\rho = 1$ as shown in Fig. 2 and only one gene is replaced/swapped.

C. Fitness Functions

A useful fitness function allows different chromosomes to be compared in a manner which clearly distinguishes the best solution from others. Since the randomness quality of the generated sequence is the most important empirical objective to be measured, the number of DIEHARD tests [7] passed is used as the fitness function. We call the chromosome passing 19 DIEHARD tests as a solution. In our context, there will be multiple solutions, so that these solutions can be further differentiated in terms of secondary objectives such as total gate count required for the CA.

A suitable chromosomal representation facilitates the efficient evaluation of chromosomes' fitness. The evaluation of different objectives requires the transformation from one representation to another; this process is facilitated by efficient algorithms

given in [9]. For example, the TR allows direct DIEHARD evaluation – the TR itself is the sequence to be tested by DIEHARD, the TTR facilitate the derivation of the function's nonlinearity and the ANFR gives the implementation details of the CA and simplifies the calculation of the CA's gate count.

IV. EXPERIMENTAL RESULTS

In this first series of experiments, the objective is to verify that the proposed chromosomal representation allows CA solutions that pass all DIEHARD tests to be located. DIEHARD testing requires a 10M byte sequence and experiments are conducted for the 24-bit CA which generates the sequence of period $2^{24} > 10\text{M byte}$. The following simple GA evolution strategy is used:

1) Create an initial population of 10 chromosomes and set the maximum of evolutions to 50.
2) The crossover range is a decreasing function of the evolution no. $\rho = \lfloor 2^{16} / \text{evolution no.} \rfloor$ while $p_m = 0.1$ for each gene.
3) After each individual's fitness is evaluated using DIEHARD [7], the top five individuals are retained and five new individuals (child) are created for the next evolution. To create each child, a random pair of parents is selected from the retained individuals.

We repeat the above procedure using 10 different initial populations and there is only one occurrence where the final evolved population does not contain a CA that passes all 19

DIEHARD tests (although it eventually did after re-running the experiment beyond 50 evolutions).

In continuing experiments, it is observed that the number of evolutions required to locate an individual passing all DIEHARD tests is slightly dependent on the crossover range and mutation probability. We noticed that changing the crossover range has a more pronounced effect on the required evolutions. This can be explained from the definition of the replacement-swapping operation to create crossover. Besides each continuous block of ρ genes to be swapped, there is a corresponding set of ρ genes which are modified and the positions of these genes are randomly distributed throughout the chromosome. Here, the crossover effect is not standalone and inherently includes mutation.

One disadvantage of the TR is that memory requirements for the GA evolution simulation program is very high. In general, TR requires $n \cdot 2^n$ bits to represent a single chromosome while the representation used in [3-5] requires only $n \cdot 2^{r+1}$ bits. For the 24-bit CA used, each chromosome requires approximately $24 \cdot 2^{24}$ bits or 384Mb of RAM. A Pentium IV 2.4Ghz with 1G memory was used for the experiments and each evolution took an average of 60 minutes to complete (the initial evolutions took longer since ρ is larger).

V. CONCLUSION

We have shown that through explicit knowledge and problem formulation, the chromosomal representation is an effective tool to demarcate the solution space efficiently. More specifically, the proposed transition representation, together with the replacement-swapping operation, facilitates the use of genetic algorithms to evolve CA that passes all DIEHARD tests. All evolved CA are ensured of desirable sequence properties because of the solution space encoded by the transition representation. This new representation offers an alternative to the popular truth table representation that is dominantly used in previous work. It allows the investigation of nonlinear sequences that have guaranteed desirable sequence properties. The truth table and algebraic normal form representations are also important intermediate steps for determining other fitness objectives such as nonlinearity, gate count, etc. As such, multi-objective GA can be considered for the further work.

REFERENCES

[1] P. Pal Chaudhuri, D. Roy Chowdhury, S. Nandi and S. Chattopadhyay, "Additive Cellular Automata Theory And Applications", Volume 1, IEEE Computer Society Press, Los Alamitos, ISBN 0-8186-7717-1, California, 1997.

[2] K. Cattell and S. Zhang, "Minimal Cost One-Dimensional Linear Hybrid Cellular Automata of Degree Through 500", Journal of Electronic Testing: Theory and Applications, Kluwer Academic Publishers, Boston, Vol. 6, pp. 255-258, 1995.

[3] M. Tomassini and M. Perrenoud, "Cryptography With Cellular Automata", Applied Soft Computing, Vol. 1, pp. 151–160, 2001.

[4] Franciszek Seredynski, Pascal Bouvry and Albert Y. Zomaya, "Cellular Automata Computations and Secret Key Cryptography", Journal of Parallel Computing, Vol. 30, Issue 5-6, Elsevier, pp 753-766, 2004.

[5] Sheng-Uei Guan and Shu Zhang, "An Evolutionary Approach to the Design of Controllable Cellular Automata Structure for Random Number Generation", IEEE Transactions on Evolutionary Computation, Vol. 7, pp. 23 -36, Feb. 2003.

[6] John A Clark and Jeremy L Jacob, "Two Stage Optimization in the Design of Boolean Functions", 5th Australasian Conference on Information Security and Privacy, LNCS 1841, Springer-Verlag, pp. 242–254, 2000.

[7] Marsaglia, "Diehard", http://stat.fsu.edu/~geo/diehard.html, 1998.

[8] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, "Handbook of Applied Cryptography", CRC Press, Boca Raton, 1997.

[9] J. Fuller, W. Millan and E. P. Dawson, "Efficient Algorithms for Analysis of Cryptographic Boolean Functions", In Proceedings of Australian Workshop on Combinatorial Algorithms, pp. 133-150, May 2002.

[10] Subhamoy Maitra and Palash Sarkar, "Cryptographically Significant Boolean Functions with Five Valued Walsh Spectra", Theoretical Computer Science, Vol. 276, pp.133-146, April 2002.

[11] D. Cvetkovic and I. Parmee, "Preference and Application in Evolutionary Multiobjective Optimization", IEEE Transactions on Evolutionary Computation, Vol. 6, pp.42–57, Feb. 2002.

[12] Rane T.D., Dewri R., Ghosh S., Mitra K. and Chakraborti N., "Modeling the Recrystallization Process Using Inverse Cellular Automata and Genetic Algorithms: Studies Using Differential Evolution", Journal of Phase Equilibria and Diffusion 26 (4), pp. 311-321, Aug 2005.

[13] Dewri R. and Chakraborti N., "Simulating Recrystallization Through Cellular Automata and Genetic Algorithms", Modeling and Simulation in Materials Science and Engineering 13 (2), pp.173-183, Mar 2005.