A Family of Controllable Cellular Automata for Pseudorandom Number Generation

Sheng-Uei Guan and Shu Zhang
Department of Electrical & Computer Engineering
National University of Singapore
10 Kent Ridge Crescents, Singapore 119260
{eleguans, engp9594}@nus.edu.sg

Abstract In this paper, we present a family of novel Pseudorandom Number Generators (PRNGs) based on Controllable Cellular Automata (CCA) ─ CCA0, CCA1, CCA2 (NCA), CCA3 (BCA), CCA4 (asymmetric NCA), CCA5, CCA6 and CCA7 PRNGs. The ENT and DIEHARD test suites are used to evaluate the randomness of these CCA PRNGs. The results show that their randomness is better than that of conventional CA and PCA PRNGs while they do not lose the structure simplicity of 1-d CA. Moreover, their randomness can be comparable to that of 2-d CA PRNGs. Furthermore, we integrate six different types of CCA PRNGs to form CCA PRNG groups to see if the randomness quality of such groups could exceed that of any individual CCA PRNG. Genetic Algorithm (GA) is used to evolve the configuration of the CCA PRNG groups. Randomness test results on the evolved CCA PRNG groups show that the randomness of the evolved groups is further improved compared with any individual CCA PRNG.

Key words: cellular automata, randomness test, pseudorandom number generator, genetic algorithm

1. Introduction

Cellular Automata (CA) was initiated in the early 1950s as a framework for modeling complex structures capable of self-reproduction and self-repair. Subsequent developments have taken place in several phases with diverse motivations. One active field is to generate pseudorandom numbers using CA. The intensive interest in this field can be attributed to the phenomenal growth of the VLSI technology that permits cost-effective realization of the simple structure of local-neighborhood CA. It has been proved in [23] that the randomness of the patterns generated by maximum-length CA is significantly better than that of LFSR (Linear Feedback Shift Register) based structures.

In the last decade, one-dimensional (1-d) CA Pseudorandom Number Generators (PRNGs) have been extensively studied [4,7,8,9,10,13,14,15,16,18,19,21]. Recent interest is more focused on two-dimensional (2-d) CA PRNGs [2,11] since it seems that their randomness is better than that of 1-d CA PRNGs. But taking into account the design complexity and computation efficiency, it is quite difficult to conclude which one is better. Compared to 2-d CA PRNGs, 1-d PRNGs are easier to implement in a large scale. In this paper, we propose a family of novel CA PRNGs that obtain the same randomness as that of 2-d CA PRNGs without losing the structure simplicity of 1-d CA PRNGs.

In the following, we first give an overview on CA and CA PRNGs in section 2. We present in section 3 the definition of eight different types of controllable cells and the properties of corresponding Controllable Cellular Automata (CCA) PRNGs. In section 4, we discuss the randomness of these CCA PRNGs and compare their randomness to that of 1-d and 2-d CA PRNGs. Section 5 presents the evolutionary approach to optimize the configuration of CCA PRNG groups which can get better randomness values than any individual CCA PRNG presented in section 3. Section 6 ends the paper with a conclusion.

2. Related Work

2.1 Cellular Automata

A cellular automaton is an array of cells where each cell is in any one of its permissible states. At each discrete time step (clock cycle) the evolution of a cell depends on its transition rule, which is a function of the present states of its *k* neighbors for a *k*-neighborhood CA. The cellular array (grid) is n-dimensional, where n=1,2,3 is used in practice. We define the state of a CA at time t to be the n-tuple formed from the states of the individual cells, $X(t)=[x_1(t),\ldots,x_n(t)]$. The next-state function of a 3-neighborhood (r=1) CA is computed as: $X(t+1)=F(X(t))=[f_1(0,x_1(t),x_2(t)),\ldots,f_i(x_{i-1}(t),x_i(t),x_{i+1}(t)),\ldots]$. When each f_i is a linear function, f is also a linear function, mapping n-tuples to n-tuples. The evolution of the i-th cell in a one-dimensional, 3-neighborhood CA can be represented as a function of the present states of the (i-1)-th, (i)-th, and (i+1)-th cells as: $x_i(t+1)=f_i(x_{i-1}(t),x_i(t),x_{i+1}(t))$, where f_i represents the transition rule for the (i)-th cell.

Some definitions to characterize the properties of CA are noted below.

Definition 1. If the rules of a CA cell involve only XOR logic, then they are called *linear rules.* Rules involving XNOR logic are referred to as *complemented rules.* In this paper, we use a combination of both linear and complemented rules. A CA having a combination of XOR and XNOR rules is called an *additive* CA.

Definition 2. If all the CA cells obey the same rule, then the CA is said to be a *uniform CA*; otherwise, it is a *non-uniform or hybrid CA.*

Definition 3. A CA is said to be a *Periodic Boundary CA* (PBCA) if the extreme cells are adjacent to each other. A CA is said to be a *null-boundary* CA if the extreme cells are only connected to its left (right) cell.

Programmable CA (PCA) is initially mentioned in [18]. It is a non-uniform CA that allows different rules to be used at different time steps on the same cell. Compared to uniform CA, PCA allows several control lines per cell. Through these control lines, different rules can be applied to the same cell at different time steps according to the rule control signals. Fig. 1 shows a programmable cell structure.

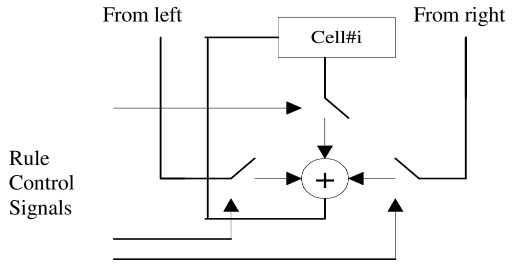


Fig. 1 A programmable cell structure

Generally, transition rule is one of the critical factors that decide the property of CA, whether it is uniform CA or PCA. Since there is a lot of work done to explore the properties of different rules, we only use those rules that have been proved to be good in random number generation in our work. Here we give the Boolean form of these rules and their numbers are given in accordance with Wolfram's convention. The following rules are either additive or linear except rule 30.

Rule 30: $x_i(t+1) = x_{i-1}(t)$ XOR ($x_i(t)$ OR $x_{i+1}(t)$)

Rule 90: $x_i(t+1) = x_{i-1}(t)$ XOR $x_{i+1}(t)$

Rule 105: $x_i(t+1) = x_i(t)$ XNOR ($x_{i-1}(t)$ XOR $x_{i+1}(t)$)

Rule 150: $x_i(t+1) = x_{i-1}(t)$ XOR $x_i(t)$ XOR $x_{i+1}(t)$

Rule 165: $x_i(t+1) = x_{i-1}(t)$ XNOR $x_{i+1}(t)$

2.2 CA Pseudorandom Number Generators

In this section, we present several 1-d and 2-d CA PRNGs proposed since the last decade. Before we proceed to introduce the generators, we first investigate what properties of CA will affect the randomness of the sequences generated by CA PRNGs.

In general, there are four aspects of CA configuration affecting the randomness:

- Boundary conditions — null boundary, periodic boundary or mirrored boundary: generally periodic boundary condition is better than null boundary condition in random number generation [17].
- Length of a CA — the total number of cells in a CA: A CA formed from N cells with a single rule generally has a cycle length much shorter than 2^N-1. As the length of the CA increases the maximum possible cycle length of the pseudorandom sequence increases.
- Initial seed — the initial state configuration in CA: Generally, the effect of initial seed on randomness is obvious. To counteract its effect, in the following work, we apply the randomness test on a set of randomly generated initial seeds instead of only one.
- Transition rule — obviously, the randomness of the sequences generated by different rules varies a lot.

2.2.1 1-d CA PRNGs

Rule-30 uniform CA has been extensively studied by Wolfram in 1986 [23]. It was the first time that computer scientists applied CA in pseudorandom number generation. Wolfram's work on rule-30 CA demonstrated its ability to produce fairly random, temporal bit sequences [20]. Wolfram also suggested that rule-30 CA can be efficiently implemented in parallel. Later, other rules were also applied in uniform CA PRNGs. Tomassini et al. concluded in [10] that according to the DIEHARD test results, rule 105 is the best, followed by rules 165, 90 and 150, with rule 30 coming in the last.

Following the idea of uniform CA PRNGs, more researchers focus their interest on non-uniform CA PRNGs since non-uniform CA PRNGs are better than uniform ones in general. The first non-uniform CA PRNG was proposed by P. D. Hortensius in 1989 [14]. This non-uniform CA uses rule 90 and 150. This CA PRNG is referred to hence as PCA 90-150. Nandi et al. showed in [18] that a PCA 90-150 built with maximal length CA configurations can generate pseudorandom patterns. Unlike uniform rule-30 CA, adjacent cells in non-uniform CA are not correlated in both time and space [14]. However, the binary sequences produced by some cells in a non-uniform CA fail some random number tests because of distribution problems. Another non-uniform CA PRNG which uses the combination of rules 30 and 45 was also proposed by P. D. Hortensius [15]. This generator can evolve to a random pattern of outputs, but its bit sequence correlation is much higher than that of the PCA 90-150 [15].

Later in 1996, Sipper and Tomassini [13] evolved a 50-cell CA with a mélange of rules 90, 150 and 165. This CA is referred to henceforth as PCA 90-165. Based on their work, Tomassini et al. [10] evolved another 50-cell CA with the rule combination 90, 105, 150 and 165 in 1999. This CA is referred to henceforth as PCA 90-105. These two 2-bit PCA are evolved using a cellular programming evolutionary algorithm while those two CA proposed by P. D. Hortensius are handcrafted. The DIEHARD test results show that these two non-uniform CA PRNGs are better than those designed by P. D. Hortensius in [14,15]. But they still cannot pass some of the tests in DIEHARD and are inferior to the classical generators.

2.2.2 2-d CA PRNGs

Work on 1-d CA PRNGs not only shows the suitability of CA in random number generation but also raises another question: is it possible to further improve the randomness of CA PRNGs? Chowdhury et al. [2] described a methodology for producing pseudorandom numbers by 2-d CA in 1994. Their results suggest that 2-d CA are superior to 1-d ones of the same size in pseudorandom number generation. Following their idea, Tomassini et al. evolved several 8×8 2-d CA PRNGs with rules 15, 63, 31 and 47 [11]. DIEHARD test results show that some of the evolved CA PRNGs can pass all the tests in DIEHARD. And based on the observation of these evolved 2-d CA PRNGs, they can handcraft even better PRNGs.

Although 2-d CA PRNGs are better than 1-d CA PRNGs in random number generation, they lose the structure simplicity in hardware design and computation efficiency in software simulation. Therefore, how to find a set of CA PRNGs with good randomness quality and the merits of 1-d CA PRNGs becomes an important problem. Under this motivation, we propose a novel CA — Controllable CA in the next section.

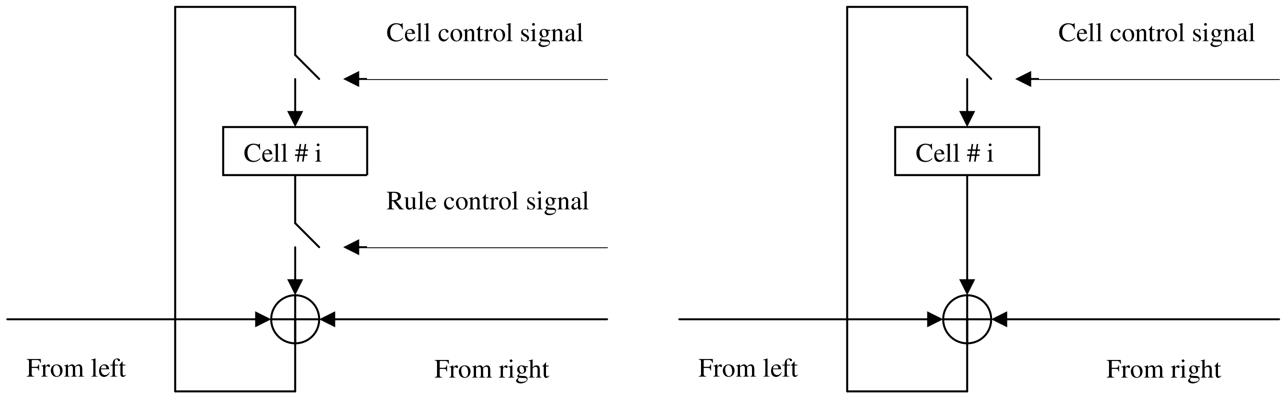
3. Controllable CA

3.1 Controllable CA

In this section, Controllable Cellular Automata (CCA) is introduced. To explain the scheme explicitly, several new concepts are defined first to identify the CCA properties.

Definition 4. A *Controllable CA* (CCA) is a CA in which the action (how the state of a cell is updated in each cycle) of some cells can be controlled via cell control signals.

Definition 5. If a cell is under the control of cell control signal, it is a *controllable cell*; otherwise it is a *basic cell*. CCA is the combination of controllable cells and basic cells. Both controllable cells and basic cells could have rule control signals. Fig. 2 shows the non-programmable/programmable controllable cell structure. In this paper, we discuss programmable controllable cells only. Therefore, controllable programmable cell is referred to henceforth as controllable cell.



(a) Programmable controllable cell **(b) Non-programmable controllable cell**

Fig. 2 A controllable cell structure

The action of a controllable cell is decided by its current cell control signal. A controllable cell can be normal (when the cell control signal is 0) or activated (when the cell control signal is 1). When the controllable cell is normal, the computation of the states of the controllable cell and its neighbors are as usual (according to the current rule control signals and the states of its neighbors). When the controllable cell is activated, the computation of the states of the controllable cell and its neighbors are specified by some predefined action. The action applied to the controllable cell and its neighbors could be different. It is the predefined action that decides the properties of controllable cells.

The structure of a CCA is shown in Fig. 3. It has L cells in total. M (M<=L) cells are controllable cells and the remaining L-M cells are basic cells. Here, all the basic cells are programmable cells. Thus, in this CCA, there are L rule control bits and M cell control bits. Compared to an L-cell PCA that has L rule control bits, the adding cost of CCA is the M cell control bits. All the CCA PRNGs discussed in this paper are based on this structure. The only difference among them is that they have different types of controllable cells. In our work, the rule (cell) control signals are generated by a CA called as rule (cell) control CA. Some of our earlier work on CCA has been published in a conference paper [19]. In the next sub-section, we will present eight different types of controllable cells and discuss the randomness of the corresponding CCA PRNGs.

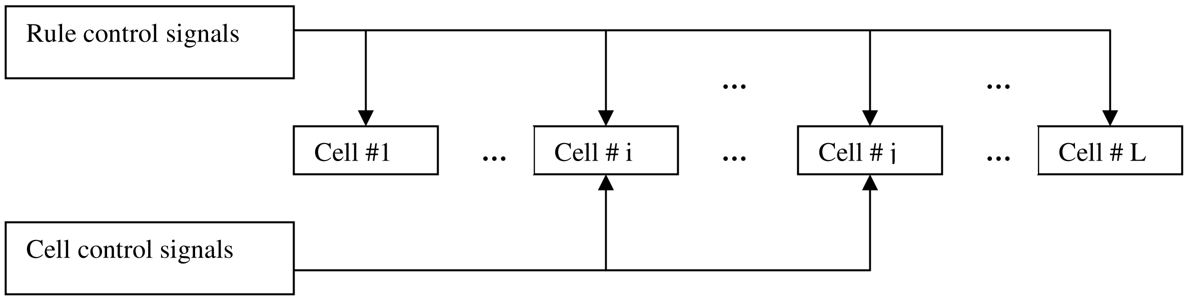


Fig. 3 The structure of a CCA

3.2 Eight Different Types of CCA

The simplest action that an activated controllable cell can do is to keep its state during the CA computation process. In the meantime, the states of its neighbors are computed as usual. This type of controllable cell is recorded as a Type 0 controllable cell. A CCA with this type of controllable cells is referred to as CCA0. If the state of an activated controllable cell is complemented and the computation of its neighbors' states is as usual, we name it as a Type 1 controllable cell. A CCA with Type 1 controllable cells is referred to as CCA1. CCA0 and CCA1 are the simplest CCA we discuss in this paper. Note that Type 0 and Type 1 controllable cells can be equivalent to 2-bit programmable cells under certain transition rules; we may question why these two types of controllable cells are proposed and how is their performance compared to 1-bit and 2-bit PCA. This question will be discussed later in section 4.2 with the aid of some randomness test results on controllable cells and basic cells. In the following, we will introduce several other different types of controllable cells first, which perform more complex actions than the type 0 and type 1 controllable cells.

A Type 2 controllable cell is defined as: when the controllable cell is activated, it keeps its latest state; while its neighbors will bypass it. This means the activated controllable cell won't be involved in the state computation of its neighbors. In this way, the neighborhood relationship is dynamically changed during the CA computation process. A CCA with this type of controllable cells is referred to as CCA2 or Neighbor-changing CA (NCA). CCA2 cannot be simulated by any PCA due to its neighbor-changing behavior.

A Type 3 controllable cell is defined as: when the controllable cell is activated, it keeps its latest state; while its neighbors will treat it as a mirror. For example, if the state of the right (left) neighbor of an activated cell is 1, then the right (left) neighbor will use its own state 1 as the state of its left (right) neighbor. In other words, we can say that the right (left) neighbor replaces the activated controllable cell with itself as its left (right) neighbor. A CCA with this type of controllable cells are referred to as CCA3 or Boundary-changing CA (BCA).

By modifying a Type 2 controllable cell slightly, we get a Type 4 controllable cell defined as the following: the right neighbor of an activated controllable cell will bypass it while the left neighbor still uses it in the CA computation. This is to break the symmetry between the right neighbor and the left neighbor. A CCA with this type of controllable cells is referred to as CCA4 or asymmetric NCA.

Except Type 1 controllable cell, activated controllable cells keep their states unchanged during the CCA computation process. It is a waste of the 1-bit memory of the controllable cell. We slightly modified Type 2 controllable cells as the following: an activated controllable cell will do the transition according to a transition rule while its neighbors will do the action as defined in Type 2. Setting the rule to 30, 105 and 165 respectively, we get Type 5, Type 6 and Type 7 controllable cells. The corresponding CCA are referred to as CCA5, CCA6 and CCA7 individually. Obviously different choice of transition rules will affect the randomness of these types of CCA. In this paper we will discuss these three rules only which are proved to be among the best additive transition rules in random number generation [14].

In the next section, we will discuss the randomness of these eight CCA PRNGs presented above and compare their randomness to PCA PRNGs and 2-d CA PRNGs.

4. The Randomness of CCA PRNGs

Before we apply the randomness tests on the controllable cells and CCA PRNGs, we firstly introduce two randomness test suites used and one randomness evaluation function. The result of this function is a real value calculated based on the randomness test results. It is used as a yardstick to compare the randomness of controllable cells and CCA PRNGs.

4.1 Introduction to Randomness Tests

There are two widely used randomness test suites — ENT and DIEHARD. The former is designed according to the criteria set by Knuth [1]; the latter is devised by G. Marsaglia [3]. A detailed introduction to these two tests is given in the appendix A. In this sub-section, we introduce how we evaluate the randomness of CCA PRNGs using the ENT test suite.

Tomassini et al. used entropy to evaluate the randomness of 2-d CA PRNGs in [11]. But our ENT test results on the CCA PRNGs show that some generators obtaining good entropy values can still fail the chi-square test. To get a better evaluation on the randomness of CCA PRNGs, we use the results of three tests (chi-square, entropy and Serial Correlation Coefficient (SCC)) instead of entropy only. We introduce a function F here to get an overall evaluation on the randomness based on the results of these three tests. Using such a “global” function, we can easily differentiate “good” random sequences from “bad” ones. Although F is empirically designed, it is only used as a guideline. DIEHARD is used to further evaluate the randomness of “good” random sequences.

As we have introduced in Appendix A, if a sequence cannot pass the chi-square test, it is thought to be non-satisfactory in randomness. That is to say, the chi-square test result is an important indication to the randomness of the sequences tested. Thus, we feel that the chi-square test is more important than entropy and SCC in evaluating the randomness of CCA PRNGs. It is difficult to decide which one is more important between entropy and SCC because they are testing different aspects of randomness. Taking into account these factors, we use a function F as follows to evaluate the overall randomness of the CCA PRNGs. We give entropy and SCC the same ratio while giving chi-square test a slightly higher ratio to emphasize it.

$$F = (\text{entropy} - 7) * 30\% + (1 - |\text{SCC}|) * 30\% + f(\text{chi-square}) * 40\% \qquad (1)$$

$$f(\text{chi-square}) = \begin{cases} 0; \text{ if chi-square} > 90\% \text{ or} < 10\% \\ 1; \text{ if } 10\% < \text{chi-square} < 90\% \end{cases}$$

The result of F is a real value between 0 and 1. We call this value as randomness value henceforth. A higher randomness value represents better randomness and the optimal value is 1. For the chi-square test, a test result falling in 10-90% is considered as random and gets 1 in the adjusted result. Otherwise a test result beyond this area is considered nonrandom and gets 0 in the adjusted result.

For the entropy test, 7 is deducted from the original entropy test results and the adjusted value most likely falls within [0, 1]. It is based on our observation from the ENT test results of CCA PRNGs under 10000 initial seeds. Generally, there is no sequence getting an entropy value less than 7. To emphasize the difference of the randomness of tested CCA PRNGs, we deduct the common value (7) they obtained from the original test results. The optimal value of the adjusted entropy test result is 1. The larger the adjusted entropy value is, the better randomness the sequence gets.

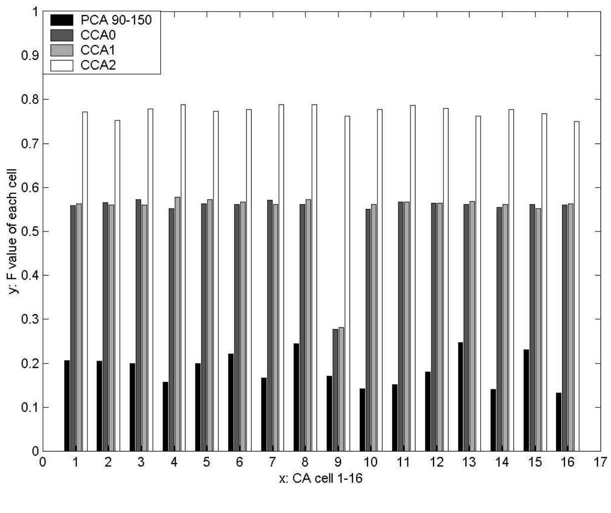
For the SCC test, the results can be positive and negative. Only the absolute value is meaningful and the sign does not affect the randomness. Generally, absolute SCC test values fall into [0, 1]. Contrary to the other two tests in which a better random sequence gets a larger adjusted result, a smaller absolute result gets a better randomness in the SCC test. To adjust an SCC value to the same direction as the other two tests, we deduct its absolute value from 1.

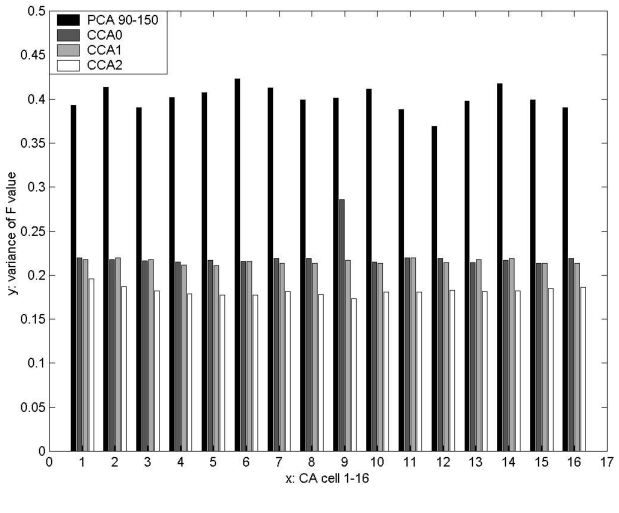
4.2 Randomness Test Results of CCA PRNGs

To compare the randomness of controllable cells and basic cells, we design a test as follows. All the CCA PRNGs have the same structure: they have 16 cells in total; the 9th cell is controllable cell and the rest are basic cells. The rule combination for CCA/PCA PRNG is 90,150. The rule control CA uses rule 105 and cell control CA used rule 165. CCA PRNGs generate random number sequences as follows: each cell generates a random bit sequence. At each time step, the state of a cell is recorded. A cell's randomness value (F value) is the ENT test result on the sequence it generated. Each CCA PRNG runs 10000 cycles to generate 16 random bit sequences. This test is repeated 10000 times. Each time a set of initial seeds (for rule control CA, cell control CA and tested CCA/PCA) is randomly generated.

An F value is calculated for each set of initial seed. Thus, for each cell, 10000 F values are obtained. The final result for each cell is the average F value and the variance of F values.

Fig. 4 shows the test results on CCA0/CCA1/CCA2 PRNGs and PCA 90-150 PRNGs. We can see that all the PCA cells obtain a randomness value about 0.2. The basic cells in CCA0 have a randomness value about 0.56 while the controllable cell gets a much lower value which is just a little higher than that of PCA cells. Note that CCA0 and CCA1 get similar test results, which means that the 'complement' action of a controllable cell is not useful to improve its randomness in this case. Although the randomness of a controllable cell is worse than that of a basic cell, it improves the overall randomness of CCA0 and CCA1. Referring to Fig. 4 (b), we can see that the variance of CCA0 and CCA1 cells, whether they are basic or controllable cells, is much lower than that of PCA cells. It means that the controllable cell can also improve the overall performance stability of CCA0 and CCA1 PRNGs.





(a) Randomness value **(b) Randomness value variance**

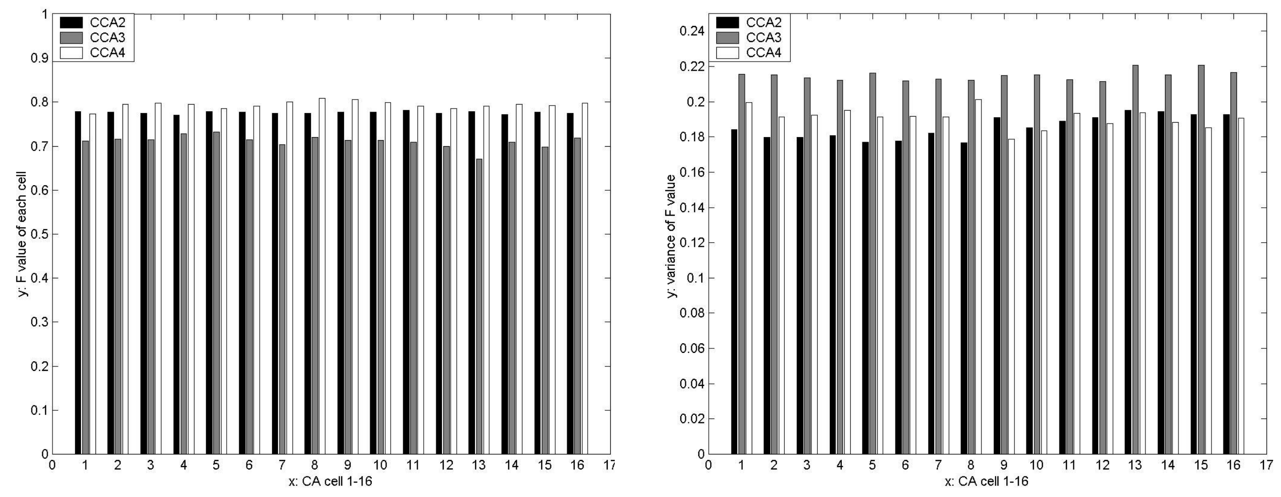
Fig. 4 Comparison of PCA/CCA0/CCA1/CCA2 randomness and variance

Notes: results are based on 10000 initial seeds.

The shortcoming of CCA0 and CCA1 is that the randomness of controllable cells is worse than that of basic cells. As shown in Fig. 4, we can see that for CCA2, the randomness of a controllable cell is similar to that of a basic cell and the F value and variance of CCA2 cells are higher than that of CCA0 and CCA1 cells. It shows that CCA2 exhibits a more stable and better randomness quality.

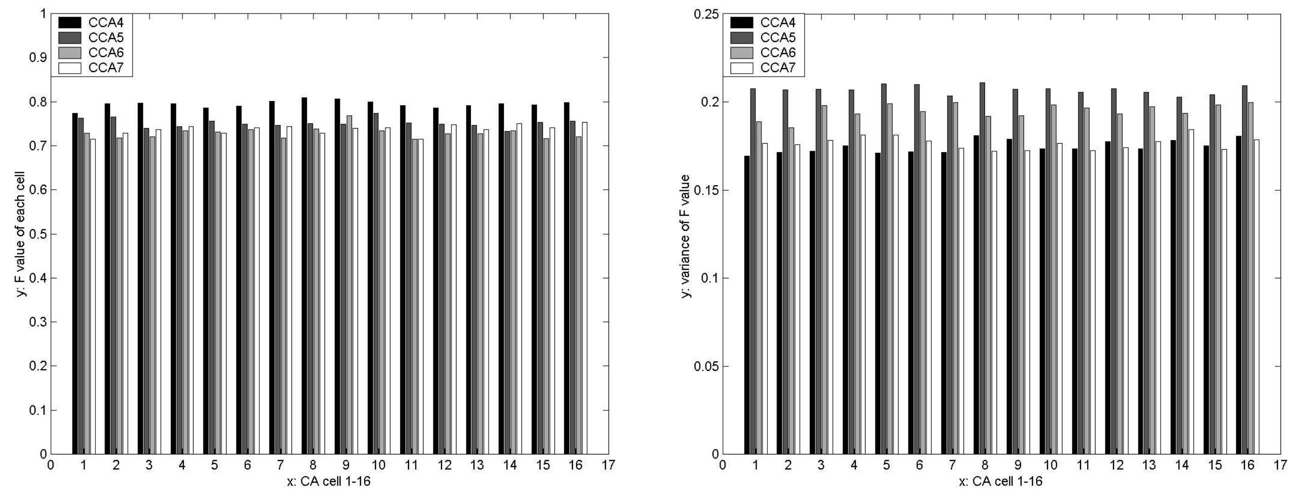
We have questioned in section 3.2 why CCA0 and CCA1 are proposed although they can be equivalent to specified 2-bit PCA. Now we have the answer inside the randomness test results from Fig. 4. The use of one single controllable cell can enhance significantly the randomness quality of CCA0 and CCA2. Also because the randomness of Type 0 and 1 controllable cells is worse than that of those basic cells in CCA0 and CCA1, we should avoid choosing Type 0/Type 1 controllable cells when we choose the output cells in CCA0/CCA1 PRNGs. It is easy to do in CCA0 and CCA1 because we can differentiate controllable cells and basic cells according to their different structures. But in 2-bit PCA in which all the cells have uniform structures, we cannot easily decide which cell should not be chosen as output cells.

The randomness test results on Type 0, Type 1 and Type 2 controllable cells tell us that the action of controllable cells decides the properties of CCA. A good ‘action’ is crucial to generate good random number sequences. In the following, we will discuss other types of controllable cells whose randomness is comparable to Type 2 controllable cells. Fig. 5 shows the randomness values of the cells in CCA2/CCA3/CCA4 PRNG. The test results show that in all these three generators, the controllable cells get similar F value as basic cells. The randomness quality of CCA3 is a little lower than that of CCA2, whether it is the F value or the variance. The F values of CCA4 cells are a little higher than those of CCA2 and the variance of CCA4 cells is also higher than that of CCA2. We may conclude that CCA2 and CCA4 are both good in random number generation and CCA4 may perform slightly better.



(a) Randomness value **(b) Randomness value variance**

Fig. 5 Comparison of CCA2/CCA3/CCA4 randomness and variance
Notes: results are based on 10000 initial seeds.



(a) Randomness value **(b) Randomness value variance**

Fig. 6 Comparison of CCA4/CCA5/CCA6/CCA7 randomness and variance
Notes: results are based on 10000 initial seeds.

Fig. 6 shows the randomness values of the cells in CCA5, CCA6 and CCA7 PRNGs. The cells in CCA5, CCA6 and CCA7 get similar F value which is lower than that of CCA4. The variance of CCA4 cells is lower than that of CCA5-7 cells too. It shows that CCA4 performs the best among all the CCA presented (CCA0-CCA7). The variance of CCA5 cells is the highest. According to the F values and variance of CCA0-CCA7 cells, we may say that CCA4 is the best among CCA0-CCA7.

		CCA0	CCA1	CCA2	CCA3	CCA4	CCA5	CCA6	CCA7
byte mode	chi-square	50%	25%	75%	50%	25%	75%	25%	75%
	SCC	0.1230	0.1245	0.0912	0.0987	**0.0901**	0.1020	0.1090	0.1087
	entropy	7.8920	7.8942	7.9226	7.9042	**7.9267**	7.9080	7.9034	7.9078
bit mode	chi-square	50%	25%	75%	50%	25%	75%	25%	75%
	SCC	0.0239	0.0246	0.0131	0.0187	**0.0114**	0.0209	0.0198	0.0187
	entropy	0.9913	0.9910	0.9973	0.9951	**0.9981**	0.9930	0.9932	0.9938

Table 1 ENT test results of CCA0-7 PRNGs in the byte & bit mode
Notes: each CCA runs 10000 cycles; ts=0; ss=1.

Till now, all the random number sequences are sequentially generated by one cell from each PCA/CCA PRNGs. But generally CCA PRNGs generate pseudorandom numbers in parallel. To evaluate the randomness of CCA PRNGs in parallel, we not only use ENT but also use a more complete test suite — DIEHARD to evaluate the randomness of them. Table 1 shows the ENT test results of CCA PRNGs both in the byte and bit mode. The structures of CCA PRNGs are the same as in Fig. 4-6. They are tested under one identical randomly generated initial seed. The site spacing parameter *ss* is the number of sites between two consecutive output cells in CA. The time spacing parameter *ts* is the number of time steps between output numbers. We can see that CCA2 and CCA4 get better results than the other generators no matter in the byte mode or the bit mode. The SCC results in the byte mode show the correlation of bytes and the SCC results in the bit mode show the correlation of bits.

Test name	L=64, P=16, ss=2, ts=1							
	CCA0	CCA1	CCA2	CCA3	CCA4	CCA5	CCA6	CCA7
1. Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
2. Runs up 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs up 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
3. 3D sphere	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
4. A parking lot	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
5. Birthday Spacing	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
6. Count the ones 1	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
7. Binary Rank 6*8	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
8. Binary Rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
9. Binary Rank 32*32	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
10. Count the ones 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
11. Bitstream test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
12. Craps wins	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
games	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
13. Minimum distance	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
14. Overlapping Permu	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
15. Squeeze	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
16. OPSO test	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
17. OQSO test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
18. DNA test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
number of tests passed	***16***	***16***	***18***	***18***	***18***	***18***	***18***	***18***

Table 2 DIEHARD test results of CCA0-7 PRNGs (L=64 cells)
Notes: P: number of output bits generated by CA in each cycle;
Count the ones 1: count the ones for specific bytes;
Count the ones 2: count the ones for a stream of bytes.

In DIEHARD, the CCA PRNGs structure and test conditions are different from those in the ENT test. Table 2 shows the DIEHARD test results of the CCA PRNGs presented in this paper. All the tested CCA PRNGs have the same structures and rule combinations. Each CCA has 64 cells. The reason why we don’t use 16 cells is that DIEHARD is very difficult to pass for small-length CA and 64 is widely used in real applications. The number of controllable cell in CCA PRNGs is kept to its minimum — 1. Only the 32th cell is controllable and all the remaining cells are basic cells. The rule combination is 90 and 150. The rule control CA uses rule 105. The cell control CA uses rule 165. It is the same as the setting in the ENT test. Site (cell) spacing and time spacing are used in PCA and CCA PRNGs to remove correlation. The random number sequences generated by CCA/PCA PRNGs are 10M bytes. The test conditions and CCA PRNGs structures described in this paragraph will be applied to all the following DIEHARD tests presented in this paper. Referring to Table 2, we can see that CCA2-CCA7 can pass the entire tests in DIEHARD. It shows that CCA PRNGs are potentially good PRNGs.

4.3 CCA PRNGs vs 1-bit PCA/2-bit PCA/2-d CA PRNGs

We have shown in the last sub-section that according to the ENT test results, CCA PRNGs are better than PCA 90-150 PRNGs. To confirm this, we use DIEHARD to compare their randomness. Table 3 shows the DIEHARD test results of CCA and PCA PRNGs under different conditions. Since CCA2 and CCA4 get the best randomness among all the CCA PRNGs, we use these two CCA PRNGs as examples in the following tests to compare the quality of the CCA PRNGs with other CA PRNGs.

We first test the CCA PRNGs without time spacing. Referring to Table 3, we can see that when *ts*=0, CCA2/CCA4 outperform PCA 90-150 PRNGs under ss=1, 2 or 3. But they still cannot pass all the tests in DIEHARD. M. Tomassini suggested in [10] that time spacing is crucial to generate a very high quality random number sequence. Our test results are in accord with his suggestion. With a time spacing of 1, CCA2/CCA4 PRNGs can pass all the tests in DIEHARD. Since under all the circumstances, CCA2/CCA4 PRNGs pass more tests than PCA 90-150 PRNGs, we can conclude with confidence that CCA are better than PCA 90-150 in random number generation.

Test name	P=32, ss=1, ts=0		P=16, ss=2, ts=0		P=8, ss=3, ts=0		P=16, ss=2, ts=1	
	PCA 90-150	CCA2 /CCA4	PCA 90-150	CCA2 /CCA4	PCA 90-150	CCA2 /CCA4	PCA 90-150	CCA2 /CCA4
1. Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
2. Runs up 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 1	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs up 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Runs Down 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
3. 3D sphere	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
4. A parking lot	Fail	Pass	Fail	Pass	Fail	Pass	Fail	Pass
5. Birthday Spacing	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
6. Count the ones 1	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
7. Binary Rank 6*8	Fail	Fail	Pass	Pass	Pass	Pass	Pass	Pass
8. Binary Rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
9. Binary Rank 32*32	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
10. Count the ones 2	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
11. Bitstream test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
12. Craps wins	Fail	Pass	Fail	Pass	Pass	Pass	Pass	Pass
throws	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
13. Minimum distance	Pass	Pass	Pass	Pass	Fail	Pass	Pass	Pass
14. Overlapping Perm.	Fail	Fail	Pass	Pass	Fail	Pass	Pass	Pass
15. Squeeze	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
16. OPSO test	Pass	Pass	Pass	Pass	Fail	Fail	Pass	Pass
17. OQSO test	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass
18. DNA test	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
Number of tests passed	***12***	***14***	***15***	***17***	***13***	***16***	***17***	***18***

Table 3 DIEHARD test results of 1-bit PCA/CCA2/CCA4 PRNGs (L=64 cells)

Notes: P: number of output bits generated by CA in each cycle;
Count the ones 1: count the ones for specific bytes;
Count the ones 2: count the ones for a stream of bytes.

Considering that CCA use more control bits than PCA, we may suspect that whether we can further improve PCA's random quality with more control bits. 2-bit PCA PRNGs, which use two control bits per programmable cell, may be a good example to be compared with CCA PRNGs. In a 2-bit PCA, 4 rules are available for each cell during CA computation. Here, we use PCA 90-105 as an example of 2-bit PCA. PCA90-105 is chosen because its performance has been proved to be good [10]. Table 4 presents the DIEHARD test results of PCA 90-150, CCA2, CCA4 and PCA 90-105 in 50 cells. It shows that with a time spacing of 1 and site spacing of 2, both CCA2/CCA4 and PCA 90-105 PRNGs can pass all the tests in DIEHARD while the PCA 90-150 PRNG fails one test. Table 4 also presents the DIEHARD test results of CA PRNGs in 16 and 32 cells. CCA2/CCA4 PRNGs also get better randomness than PCA 90-150 PRNGs. When the length of CA is 32, CCA2/CCA4 PRNGs can pass one more test than the 2-bit PCA 90-105 PRNG.

Test name	L=50, P=16, ss=2, ts=1			L=32, P=16, ss=1, ts=1			L=16, P=8, ss=1,ts=1		
	1-bit PCA 90-150	2-bit PCA 90-105	CCA2/ CCA4	1-bit PCA 90-150	2-bit PCA 90-105	CCA2 /CCA4	1-bit PCA 90-150	2-bit PCA 90-105	CCA2 /CCA4
1. Overlapping sum	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
2. Runs up 1	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Pass	Pass
Runs Down 1	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
Runs up 2	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
Runs Down 2	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
3. 3D sphere	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
4. A parking lot	Pass	Pass	Pass	Fail	Fail	Pass	Fail	Pass	Pass
5. Birthday Spacing	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
6. Count the ones 1	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
7. Binary Rank 6*8	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
8. Binary Rank 31*31	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
9. Binary Rank 32*32	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
10. Count the ones 2	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
11. Bitstream test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
12. Craps wins	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Pass	Pass
throws	Pass	Pass	Pass	Fail	Pass	Pass	Fail	Fail	Fail
13. Minimum distance	Pass	Pass	Pass	Fail	Pass	Pass	Fail	Fail	Fail
14. Overlapping Permu	Fail	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail
15. Squeeze	Pass	Pass	Pass	Fail	Pass	Pass	Fail	Fail	Fail
16. OPSO test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
17. OQSO test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
18. DNA test	Pass	Pass	Pass	Fail	Fail	Fail	Fail	Fail	Fail
Number of tests passed	***17***	***18***	***18***	***9***	***12***	***13***	***0***	***3***	***3***

Table 4 DIEHARD test results of 1-bit PCA 90-150/2-bit PCA 90-105/CCA2/CCA4 PRNGs

Notes: P: number of output bits generated by CA in each cycle;
Count the ones 1: count the ones for specific bytes;
Count the ones 2: count the ones for a stream of bytes.

M. Tomassini et al. evolved several 2-d CA PRNGs in [11]. They showed that some of the evolved 8×8 2-d CA PRNGs could pass all the tests in DIEHARD. Table 5 shows the DIEHARD test results of theirs and ours. We can see that CCA2/CCA4 PRNG with a time spacing of 1 can pass all the tests in DIEHARD too. Thus, we may say that according to the DIEHARD test results, CCA PRNGs can compete with 2-d CA PRNGs.

Test name	1-d CCA2/CCA4 L=50, P=16,ss=2,ts=1	2-d CA PRNG (8×8) Tomassini et al.
1. Overlapping sum	Pass	Pass
2. Runs up 1	Pass	Pass
Runs Down 1	Pass	Pass
Runs up 2	Pass	Pass
Runs Down 2	Pass	Pass
3. 3D sphere	Pass	Pass
4. A parking lot	Pass	Pass
5. Birthday Spacing	Pass	Pass
6. Count the ones 1	Pass	Pass
7. Binary Rank 6*8	Pass	Pass
8. Binary Rank 31*31	Pass	Pass
9. Binary Rank 32*32	Pass	Pass
10. Count the ones 2	Pass	Pass
11. Bitstream test	Pass	Pass
12. Craps wins	Pass	Pass
throws	Pass	Pass
13. Minimum distance	Pass	Pass
14. Overlapping Permu	Pass	Pass
15. Squeeze	Pass	Pass
16. OPSO test	Pass	Pass
17. OQSO test	Pass	Pass
18. DNA test	Pass	Pass
Number of tests passed	***18***	***18***

Table 5 DIEHARD test results of 2-d CA/1-d CCA2/CCA4 PRNGs

Notes: P: number of output bits generated by CA in each cycle;
Count the ones 1: count the ones for specific bytes;
Count the ones 2: count the ones for a stream of bytes.

Except statistical tests, cycle length (the length of a CA's state cycle) is also important to determine the suitability of a CA for random number generation. Tomassini et al. calculated the cycle length of their evolved 2-d 4*4 CCA PRNGs over 20 initial seeds [11]. To facilitate the comparison with their results, we test CCA PRNGs over 20 initial seeds too. Considering the fairness for comparison and computation load of calculating the cycle length of a CA with large size, we only test small-size (L=16) CA here. We realize that the length of CA and the number of initial seeds tested may not be large enough to get the cycle length of CA. But the results presented here are meaningful for comparison purpose.

Table 6 shows the cycle lengths of 1-bit PCA 90-150, 2-bit PCA 90-105, CCA0, CCA2, CCA4 and 2-d CA PRNGs. The results show that the average cycle length of PCA 90-150 is the smallest. The average cycle length of CCA0 is slightly greater than PCA 90-105, but less than CCA2 and CCA4. The average cycle length of 2-d CA is greater than CCA0 but less than CCA2 and CCA4. It means that CCA PRNGs can be better than or comparable to 2-d CA PRNGs.

Type(No. of cells)	Avg. cycle length	Max cycle length	Max/Avg.	Log_2 (Avg. cycle)
PCA 90-150 (16)	2521	65536	26.0	11.3
PCA 90-105 (16)	2943	65536	25.8	11.4
CCA0 (16)	3179	65536	20.6	11.6
CCA2 (16)	15411	65536	4.25	13.96
CCA4 (16)	**15567**	65536	**4.21**	**13.97**
2-d CA (4×4)	4778	65536	13.72	12.22

Table 6 Average and maximum cycle lengths of PCA/CCA PRNGs
Notes: results are based on 20 initial seeds.

5. Evolutionary Approach to Groups of CCA PRNGs

We have discussed the randomness of a set of CCA PRNGs in the last two sections. Randomness test results show that except CCA0 and CCA1 PRNGs, these CCA PRNGs' randomness is in the same range. In this section, we discuss how to integrate these PRNGs into CCA PRNG groups to generate random number sequences with better randomness quality. We select all the CCA PRNGs except

CCA0 and CCA1 as the basic generators to be used in a CCA PRNG group. These CCA PRNGs are: CCA2 (PRNG 0), CCA3 (PRNG 1), CCA4 (PRNG 2), CCA5 (PRNG 3), CCA6 (PRNG 4) and CCA7 (PRNG 5). A simple function ─ MOD is used to integrate the sequences generated by the PRNGs in the group. A new sequence is generated as the output of the CCA PRNG group by applying MOD to the sequences generated by the generators in this CCA PRNG group.

In each CCA PRNG group, each basic generator can either be used or not used. The objective of our study is to find which generators will be present in the evolved CCA PRNG groups and their distributions in the results. We know the effect of initial seeds on the randomness of CCA PRNGs. To find the distribution of good CCA PRNG groups for a wide range of initial seeds, we search under T (T=100) randomly generated initial seeds. The search space is 64 under one initial seed. It is so small that even exhaustive search will work well here. Yet taking into account that the searching process will repeat T times and the software simulation on CCA transitions is quite time-consuming, we use Genetic Algorithm (GA) in our work to evolve CCA PRNG groups under each initial seed. Another reason that we use GA here is that this method is scalable. We consider six CCA PRNGs here only, but we may have more variations of CCA developed in the future. A CCA PRNG group may use more generators than six as presented here. Using GA, the algorithm can be easily modified and used in the future work.

To simplify the evolution process, we do not evolve the structure of any individual CCA PRNG here. All the CCA PRNGs have the same structures. Each CCA has 16 cells where the 9th cell is a controllable cell and the remaining are basic cells. The rule combinations are the same as the setting in the previous ENT and DIEHARD tests. Each PRNG generates an 8-bit integer as output number every cycle and runs C cycles to generate a random number sequence. C is set to 10000 here.

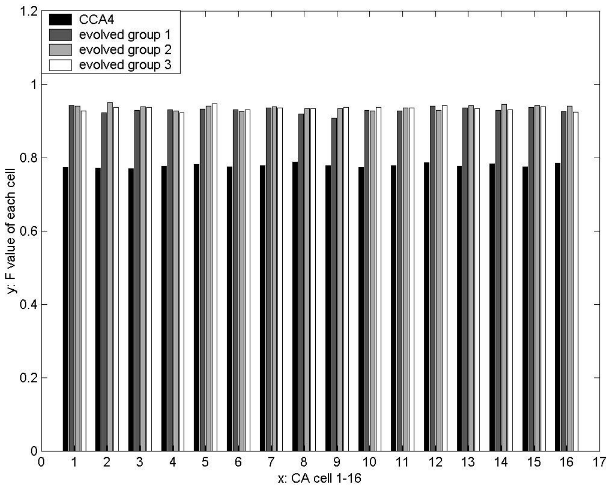
Algorithm 1: Evolving CCA PRNG groups under T initial seeds (T=100)

```
for t =1 to T do
    //initialization
     randomly generate the initial population with a fixed size P (P=8);
    for i = 0 to 5 do
        initialize PRNG i with a randomly generated initial seed;
        PRNG i runs 10000 cycles to generate a random number sequence i;
    end for (i)
    //evolution
    while (stopping criteria is not true)
      //Fitness calculation
      for m = 1 to P do
        calculate each chromosome's fitness: the sequences of the selected PRNGs are
        integrated using the MOD function to generate a new sequence, F value of this
        sequence is the fitness value of the chromosome;
      end for (m)
      //crossover & mutation
      scale fitness value using the windowing method;
      roulette-wheel select parent chromosomes, do 1-point crossover to generate  8 child
      chromosomes;
      do mutation on the child chromosomes, mutation rate is 0.01;
      //selection
      calculate child chromosomes' fitness;
      copy the best P* RATE (RATE=0.5)  parent chromosomes to the next generation;
      copy the best P-P*RATE child chromosomes to the next generation;
    end while (evolution)
    record the best chromosome's fitness value and configuration;
end for
```

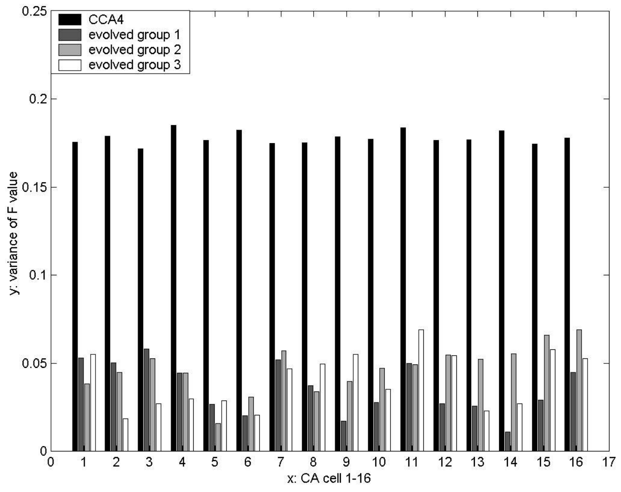
The evolutionary approach is presented in Algorithm 1. Each chromosome has six bits to encode the configuration of a CCA PRNG group with each bit identifying one CCA PRNG from PRNG 0 to PRNG 5 in sequence. 1 means this PRNG is included in this CCA PRNG group; 0 means it is not included. Population size P is set to 8 because the search space is small (2^6) for one test. The fitness of a CCA PRNG group is the F value (introduced in section 4.1) of the sequence it generated. The stopping criteria is maximal stagnation times, which is set to 50. If the best chromosome's fitness has not been improved for 50 generations continuously, the evolution process will stop. Crossover rate is set to 1.0 and we use one-point crossover here since the length of chromosomes is small. Mutation rate is set to 0.01 for all the bits in the chromosome. The selection RATE is set to 0.5.

The statistics of the evolution results under 100 initial seeds is as follows. The distribution of each individual CCA PRNG being selected in the best chromosomes is: PRNG 0: 51; PRNG 1: 50; PRNG 2: 52; PRNG 3: 49; PRNG 4: 45; PRNG 5: 45. The result shows that no CCA PRNG is predominant and each CCA PRNG has similar possibility to be used in the evolved CCA PRNG groups. Our test is based on 100 initial seeds which may be not large enough to get a final conclusion but we think it is a valuable indication at least.

The evolution result for each initial seed is a 6-bit chromosome indicating which generators are used in the corresponding CCA PRNG group. We present some evolved CCA PRNG groups (evolved group 1 to 3, chosen from the 100 evolved groups) in Appendix B as examples. Fig. 7 shows the randomness values of evolved CCA PRNG groups and their randomness variance based on 10000 initial seed runs. The test condition is the same as the previous ENT test described in section 4.2. We can see that the evolved three CCA PRNG groups get a randomness value close to 1 and the variance of the randomness values is around 0.05 while the best variance obtained by individual CCA PRNG is around 0.17. The highly decreased variance of evolved CCA PRNG groups means that the performance stability of evolved CCA PRNG is better than each individual CCA PRNG.



(a) Randomness value



(b) Randomness value variance

Fig. 7 Comparison of CCA4/evolved groups 1-3 in randomness and variance
Notes: results are based on 10000 initial seeds.

Table 7 shows the DIEHARD test results of the evolved CCA PRNG groups (evolved group 1-3). We can see that all PRNGs except CCA2 and CCA4 fail all the tests in DIEHARD, while the evolved CCA PRNG groups (1 to 3) can pass 13 tests. It is evident that the randomness of the evolved CCA PRNG groups is highly improved. Table 8 shows the cycle lengths of these evolved CCA PRNG groups. The results are calculated as average values over 20 random initial seeds. The results show that the average cycle length of each evolved CCA PRNG group is greater than that of any individual CCA PRNG. And all the tested CCA PRNG groups get a cycle length value close to the maximum value. It is highly improved even compared to the value got by the best individual CCA PRNG. This matches with the conclusion we have derived from the ENT and DIEHARD tests that the randomness of each evolved CCA PRNG group exceeds that of any individual CCA PRNG.

Test name	L=16, P=8, ss=1,ts=0						Evolved Group1	Evolved Group 2	Evolved Group 3
	CCA 2	CCA 3	CCA 4	CCA 5	CCA 6	CCA 7			
1. Overlapping sum	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
2. Runs up 1	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
Runs Down 1	Fail	Fail	Pass	Fail	Fail	Fail	Pass	Pass	Pass
Runs up 2	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
Runs Down 2	Fail	Fail	Pass	Fail	Fail	Fail	Pass	Pass	Pass
3. 3D sphere	Pass	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
4. A parking lot	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
5. Birthday Spacing	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
6. Count the ones 1	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
7. Binary Rank 6*8	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
8. Binary Rank 31*31	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
9. Binary Rank 32*32	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
10. Count the ones 2	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
11. Bitstream test	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
12. Craps wins	Fail	Fail	Pass	Fail	Fail	Fail	Pass	Pass	Pass
throws	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
13. Minimum distance	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
14. Overlapping Permu	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
15. Squeeze	Fail	Fail	Fail	Fail	Fail	Fail	Pass	Pass	Pass
16. OPSO test	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
17. OQSO test	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
18. DNA test	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
Number of tests passed	***1***	***0***	***1***	***0***	***0***	***0***	***13***	***13***	***13***

Table 7 DIEHARD test results of individual CCA PRNGs and three evolved CCA PRNG groups
Notes: P: number of output bits generated by CA in each cycle;
Count the ones 1: count the ones for specific bytes;
Count the ones 2: count the ones for a stream of bytes.

	CCA2	CCA3	CCA4	CCA5	CCA6	CCA7	Evolved Group1	Evolved Group 2	Evolved Group 3
Avg. cycle length	15411	8602	15567	9179	9943	8582	50203	52107	50890
Max Cycle length	65536	65536	65536	65536	65536	65536	65536	65536	65536
Max/Avg.	4.25	7.62	6.55	7.14	6.59	7.64	1.31	1.26	1.29
Log_2 (Avg. cycle)	13.96	13.1	13.34	13.21	13.33	13.11	15.67	15.72	15.69

Table 8 Cycle lengths of individual CCA PRNGs and evolved CCA PRNG groups
Notes: results are based on 20 initial seeds for 16-cell CCA PRNGs.

6. Conclusion

In this paper, we have discussed several CCA PRNGs and compared them with 1-bit/2-bit PCA PRNGs and 2-d CA PRNGs. We find that CCA are better in random number generation than PCA while not losing structure simplicity. They can compete with 2-d CA PRNGs while their computation efficiency and wiring cost are less than that of 2-d generators. We have also compared the randomness of several different types of CCA PRNGs. CCA0 and CCA1 have the simplest configurations but their randomness is the worst. CCA4 get the best randomness quality among all the tested generators considering the results from DIEHARD and cycle length. All of them can pass the entire tests in DIEHARD with proper site and time spacing. Further, these CCA PRNGs are evolved together to generate better randomness sequences. Evolution results show that each CCA PRNG has similar possibility to be integrated with other CCA PRNGs as a group. The randomness of the evolved CCA PRNG groups is better than any individual CCA PRNG and their randomness is more stable under different initial seed settings.

In addition to random number generation, CCA may be used in other applications such as BIST (Built-In Self-Test) or error correcting codes due to their suitability in VLSI design. Also, we may use CCA in stream cipher and private/public cipher systems. Moreover, the usage of controllable cells in CCA makes them possible for some applications where conventional CA cannot work. For example, if a CA cell is malfunctioning, a CCA2 with neighbor changing property can easily bypass this node without bringing the system down.

REFERENCES

[1] D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd ed. Reading, Mass., Addison-Wesley, 1998.

[2] D. R. Chowdhury, I. S. Gupta and P. Pal Chaudhuri, A class of two-dimensional cellular automata and applications in random pattern testing, Journal of Electrical Testing: Theory and Applications, Vol. 5, 1994, pp. 65-80.

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

[4] I. Kokolakis, I. Andreadis and Ph. Tsalids, Comparison between cellular automata and linear feedback shift registers based pseudo-random number generators, Microprocessors and Microsystems, Vol. 20, 1997, pp. 643-658.

[5] Jason D. John and James A. Reggia, Automatic Discovery of self-replicating structures in cellular automata, IEEE Trans. on Evolutionary Computation, Vol. 1, No. 3, Sep. 1997, pp. 165-178.

[6] Justin Werfel, Melanie Mitchell and James P. Cruchfield, Resource Sharing and Coevolution in Evolving Cellular Automata, IEEE Trans. on Evolutionary Computation, Vol. 4, No. 4, Nov. 2000, pp. 388-393.

[7] M. Matsumoto, Simple cellular automata as pseudorandom m-sequence generators for built-in self-test, ACM Trans. on Modeling and Computer Simulation, Vol. 8, No. 1, 1998, pp. 31-42.

[8] M. Mihaljevic, Security examination of a cellular automata based pseudorandom bit generator using an algebraic replica approach, In Proceedings of Applied Algebra, Algorithms and Error Correcting Codes, Lecture notes in Computer Science, Vol. 1255, 1997, pp. 250-262.

[9] M. Mihaljevic and Hideki IMAI, A family of fast keystream generators based on programmable linear cellular automata over GF(q) and time-variant table, IEICE Trans. Fundamentals, Vol. E82-A, No. 1, 1999, pp. 32-39.

[10] Marco Tomassini, Moshe Sipper, Mose Zolla and Mathieu Perrenoud, Generating high-quality random numbers in parallel by cellular automata, Future Generation Computer Systems, Vol. 16, 1999, pp. 291-305.

[11] Marco Tomassini, Moshe Sipper and Mathieu Perrenoud, On the generation of high-quality random numbers by two-dimensional cellular automata, IEEE Trans. Comput., Vol. 49, 2000, pp. 1146-1151.

[12] Moshe Sipper, Evolution of Parallel Cellular Machines, The Cellular Programming Approach, Springer-Verlag, Berlin-Heidelberg, 1997.

[13] Moshe Sipper and Marco Tomassini, Generating parallel random number generators by cellular programming, International Journal on Modern Physics, Vol. 7, No. 2, 1996, pp.181-190.

[14] P. D. Hortensius, R.D. Mcleod, and H.C. Card, Parallel random number generation for VLSI system using cellular automata, IEEE Trans. on Computers, Vol. 38, 1989, pp. 1466-1473.

[15] P. D. Hortensius, R. D. Mcleod, Werner Pries, D. Michael Miller and H. C. Card, Cellular automata-based pseudorandom number generators for built-in self-test, IEEE Transactions on Computer-Aided Design, Vol. 8, No. 8, 1989, pp. 842-859.

[16] P. P. Chaudhuri, D. R. Chowdhury, S. Nandi, and S. Chattopadhyay, Additive cellular automata: theory and applications, Vol. 1, Los Alamitos, Calif., IEEE CS Press, 1997.

[17] Palash Sarkar, A brief history of cellular automata, ACM Computing Surveys, Vol. 32, No. 1, 2000, pp. 80-107.

[18] S. Nandi, B. K. Kar, and P. Pal Chaudhuri, Theory and applications of cellular automata in cryptography, IEEE Trans. on Computers, 43, 1994, pp. 1346-1357.

[19] Sheng-Uei Guan and Shu Zhang, "An Encryption Method based on Dynamic Cellular Automata", Proceedings the International ICSC Congress on Intelligent Systems and Applications, University of Wollongong, Australia, # 1514-074(CD number: ISBN 3-906454-24-X), December 12-15, 2000.

[20] S. Wolfram, Theory and Applications of Cellular Automata: Including Selected Papers 1983-1986, World Scientific publishing Co., Inc., River Edge, N. J. 1986.

[21] W. Pries, A. Thanailakis, and H.C. Card. Group properties of cellular automata and VLSI applications, IEEE Trans. on Computers, Vol. C-35, 1986, pp. 1013-1024.

[22] ENT test suite, http://www.fourmilab.ch/random

[23] S. Wolfram, Cryptography with cellular automata, Advances in Cryptography: Proceedings of CRTPTO 85, Lecture Notes in Computer Science, Vol. 218, 1985, pp. 429-432.

APPENDIX A

1. ENT Test

ENT is a Pseudorandom Number Sequence Test Program, which applies various tests to sequences of bytes stored in files and reports the results of those tests [22]. This program is useful for evaluating pseudorandom number generators for encryption. ENT performs a variety of tests on the input stream of bytes in *in_file* and produces output as follows on the standard output stream:

- Entropy: the information density of the contents of the file, expressed as a number of bits of character. The optimal value is 8. Larger value means better randomness.
- Chi-square Test: the most commonly used test for the randomness of data, sensitive to errors in pseudorandom sequence generators. This test is calculated for the stream of bytes in the file and expressed an absolute number and a percentage that indicates how frequently a truly random sequence would exceed the value calculated. "Good" results are between 10%-90%, with extremities on both sides representing non-satisfactory random sequences.
- Serial Correlation Coefficient (SCC): measures the extent to which each byte in the file depends upon the previous byte. For random sequences, this value should be close to 0. Whether the value is positive or negative does not affect the randomness and smaller absolute value means better randomness.

2. DIEHARD

DIEHARD seems to be the most powerful test for randomness. Generally, a PRNG which can pass DIEHARD can be considered as good. The DIEHARD battery of tests consists of 18 different, independent statistical tests. Results of tests are so called "P-value" which is a real number between 0 and 1. For any given test, a smaller P-value means a better test result with the exception that a P value less than 0.025 or larger than 0.975 means that the PRNG has failed the test at the .05 level. A complete description of all the tests in DIEHARD is available in [3].

APPENDIX B

PRNG 0: CCA2; PRNG 1: CCA3; PRNG 2: CCA4;

PRNG 3: CCA5; PRNG 4: CCA6; PRNG 5: CCA7;

<u>The configuration of Evolved Group 1: 100101</u>

PRNG 0's F value: 0.935413

PRNG 1's F value: 0.928216

PRNG 2's F value: 0.947436

PRNG 3's F value: 0.936831

PRNG 4's F value: 0.922905

PRNG 5's F value: 0.926204

Evolved Group 1's F value: 0.952342

<u>The configuration of Evolved Group 2: 110011</u>

PRNG 0's F value: 0.946413

PRNG 1's F value: 0.942163

PRNG 2's F value: 0.947930

PRNG 3's F value: 0.940311

PRNG 4's F value: 0.932789

PRNG 5's F value: 0.938680

Evolved Group 2's F value: 0.957789

<u>The configuration of Evolved Group 3: 011010</u>

PRNG 0's F value: 0.946262

PRNG 1's F value: 0.942324

PRNG 2's F value: 0.946734

PRNG 3's F value: 0.939023

PRNG 4's F value: 0.938903

PRNG 5’s F value: 0.947892

Evolved Group 3’s F value: 0.955089