Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (22): 4071-4075, 2009
ISSN 1812-5654

Innovative Approach to Generate Uniform Random Numbers Based on a Novel Cellular Automata

[1]R. Ayanzadeh, [2]K. Hassani, [1]Y. Moghaddas, [1]H. Gheiby and [3]S. Setayeshi
[1]Department of Computer, Islamic Azad University,
Qazvin Branch, Qazvin, Iran
[2]Department of Mechatronics, Khajeh-Nasir University of Technology, Tehran, Iran
[3]Department of Nuclear Engineering, Amirkabir University of Technology, Tehran, Iran

Abstract: Generating random numbers plays outstanding role in computer simulations. Most applications demand for uniform random numbers more than other distributions. In this study concept of two-layer cellular automata and a novel neighborhood structure are introduced. According to these concepts, a novel approach for uniform random number generating is proposed. First layer consists of binary cellular automata which are responsible for activation and inactivation of cells in next layer. A cellular automaton with integer values is used for second layer. Interaction between layers of represented cellular automata leads to a dynamic and complex behavior of proposed model. To evaluate the quality of proposed model, several simulations were implemented. Results prove that two-layer cellular automata generate better uniform random numbers in comparison with MATLAB. Simulation of innovative RNG based on cellular automata, shows promising results, which encourage further research with the proposed techniques in this and related domains.

Key words: Cellular automata, uniform random variable, random number generators

INTRODUCTION

Dynamics of complex systems lead many processes in real world to be assumed stochastic and ambiguous (Bar-Yam, 1997). Due to this motivation in recent decades, scientists has paid attention to computer based random number generating in complex system simulations and attracted many researchers to introduce and develop these methods (Ayanzadeh *et al*., 2008; Moghaddas *et al*., 2008; Bar-Yam, 1997; Banks *et al*., 2004).

Lottery, computer games, cryptography, calculation with Monte Carlo method, computer simulations, operational research and most of intelligent optimization algorithms such as genetic algorithm, particle swarm optimization, tabu search and other Meta-heuristics are some applications of random number generators (Jang *et al*., 1997; Banks *et al*., 2004; Viega, 2003; Kalos, 2007). Random numbers are generally classified to 3 categories as below:

Truly random numbers: In this category, all numbers have equal probability to be generated. This class is not periodic and the numbers don't follow any pattern. In addition, truly random numbers are not generated by specific algorithm and prediction of next element of sequence is not possible. Indeed there is no correlation among these kinds of random numbers (Kohlbrenner and Gaj, 2004).

Pseudo random numbers: Pseudo random numbers are generated by specific algorithms and it is possible to predict some subsequences by considering generated trajectory. To start the algorithm it is needed that some of parameters be initialized. One of the most obvious problems about this category is existence of periods and specific patterns in sequences (Viega, 2003).

Quasi random numbers: In fact quasi random numbers are sequences of nonrandom numbers which are shuffled to be seemed random. Thus these types of random numbers are so suitable for calculation with Monte Carlo method (Lecuyer, 2003; Chen and Markel, 2005).

It is clear that random number generators must be adaptable with various statistical distributions due to application (e.g., uniform, normal, exponential, Poisson, Erlang). Uniform random numbers are applied in vast variety of applications. According to outstanding role of

Corresponding Author: Ramin Ayanzadeh, P.O. Box 51745/534, Tabriz, Iran Tel: +98 912 677 4147

uniform random numbers in computer simulations, represented approach in this study is proposed to satisfy these requirements.

High period, low computational space and time order for random number generation and low correlation among random numbers are some of important factors which determine the performance of a typical random number generator (Ayanzadeh *et al*., 2008).

RANDOM NUMBER GENERATORS

In 1927, Tippet designed table of forty thousand random numbers to use in various applications. One hundred thousand of random numbers were generated in a table designed by Kendall in 1939. Smith followed Kendall's study and designed mechanical random generator device in 1955. The exciting point about these tables is that they were filled without any specific algorithm. In 1951 Neumann proposed a computational method (however this method had low performance). In recent decades several algorithms were developed for random number generation as below (Ayanzadeh *et al*., 2008; Moghaddas *et al*., 2008; Hortensius *et al*., 1989).

Linear congruential generators: Linear Congruential methods use specific algorithm to generate random numbers. These algorithms are iterative and initial state is needed to start the algorithm. A sample of these algorithms is indicated in Eq. 1:

$$X_n = (aX_{n-1} + c) \mod m \quad (1)$$

where, X_{n-1} is the generated random number in previous iteration a and c are constant coefficients, m is congruential module (one unit more than maximum allowed random number) and X_n is the output of algorithm. In this method generated random number extremely depends on its previous value. Maximum period of m this algorithm is m (Chen and Merkel, 2005).

Multiple recursive generators: Multiple recursive generators are like linear congruential generator, but this method use k random numbers from previous iterations. A multiple recursive generator is indicated in Eq. 2.

In Eq. 2, a_i are constant coefficients of algorithm, $X_{n-i}$is output of algorithm in $(n-i)^{th}$ iteration and m is congruential module (one unit more than maximum allowed random number). The advantage of this method is that maximum period of algorithm is 2^m which is much more than period of Linear congruential method (Chen and Merkel, 2005).

$$X_n = \sum_{i=1}^{k} a_i X_{n-i} \mod m, \quad i=1,2,....,k \quad (2)$$

Lagged fibonacci generators: Lagged Fibonacci generators are a special case of famous Fibonacci sequence which use two outputs of previous iterations. Eq. 3 indicates the general form of this method.

$$X_n = (X_{n-l} + X_{n-k}) \mod m, \quad 0 < k < l \quad (3)$$

where, m, X_{n-k} and X_{n-l} are same with these parameters in multiple recursive generator method. k and l are the indexes of numbers which were generated in previous iterations. Performance of algorithm depends on selection of these values (Chen and Merkel, 2005).

Summation operator in Eq. 3 can be changed by any other operator (e.g., subtraction operator). Moreover it is possible to use binary logic operators to generate random bits. In this case if the operator is exclusive or (XOR) the method will be called transfer register generator thus the congruential operator will be neglected from equation and Eq. 3 will be changed to Eq. 4:

$$X_{n+1} = X_{n-p} \oplus X_{n-q} \quad (4)$$

Output of Eq. 1 up to Eq. 3 will be random numbers between zero and m.

Blum blum shub random generator: This generator was introduced by Blum and his team but due to slow functionality of this method it was never used in computer simulations. This method is widely used in cryptography. By using this method, random numbers will be generated via Eq. 5:

$$X_{n-1} = (X_x)^2 \mod m \quad (5)$$

where, m is congruential module and usually is considered as production of two big prime numbers (Chen and Merkel, 2005).

CELLULAR AUTOMATA

Cellular Automata (CA) are discrete computational models that contain networks of completely same cells which have interaction with together within a neighborhood structure. Various neighborhood structures are proposed till now. Some of the most popular models are: Neumann, Moore, Cole and Smith which are shown in Fig. 1. In Fig. 1, it is assumed that neighborhood radius equals one (Ayanzadeh *et al*., 2008; Bar-Yam, 1997; Sarkar, 2000).

Cells state (value) is selected from a finite set. These values are changing synchronously in iterations by using of some transition rules which are same for all cells. Next states will be determined according to current values of cells and current values of neighbors (Sarkar, 2000).

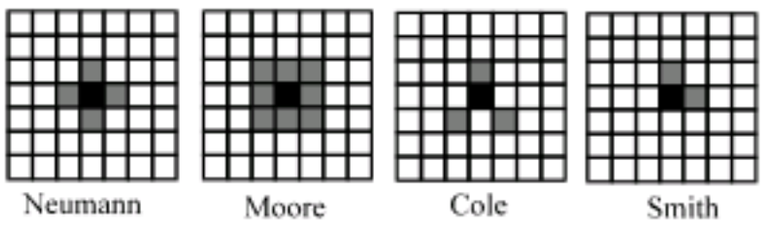

Fig. 1: Common neighborhood structures in cellular automata

Binary cellular automata are one of the most common simulation tools where each cell can be stated as zero or one. Transition rules are determined by Boolean algebra rules (AND, OR and NOT). Wolfram (1986) proposed to use decimal value of bit sequence of next state to name the rules.

Some of the most useful Wolfram transition rules in linear binary cellular automata are shown in Fig, 2 where first row is current states of left neighbor, the cell and right neighbor, respectively. Next state of cell is indicated in other rows by using of specified rules. Using transition rules in Fig. 2 and starting from a random configuration leads to generate pseudo random bits. Locality of rules leads to generate pseudo random bits with desirable period (Wolfram, 1986).

TWO-LAYER CELLULAR AUTOMATA FOR RANDOM NUMBER GENERATING

Generating sequence of binary bits and combining these bits is one of the most popular methods used in cellular automata based random number generators. It is clear that quality of generated random numbers in such methods depends on quality of sequence of random bits. Binary cellular automata will generate high period pseudo random bits by using wolfram transition rules 30, 90, 105, 110 and 165 in isolated or hybrid manner.

Framing the sequence of generated random bits-base mapping-will cause to appearance of various patterns in final sequence of random numbers. Obviously appearance of patterns among sequences of random numbers means low quality of random number generator. It is possible to compensate this problem by using parallel cellular automata to some extent. However, range of generated random numbers can demand large number of needed bits. In this case using independent cellular automata per bit will extremely increment consumption of memory and time order.

Mismatching between range of random numbers before and after base mapping is the other problem that will cause improper results. In binary numerical systems, a binary number with length n envelopes range of zero up to 2^n-1. Thus if it is impossible to map the desired range of random numbers to such range, mapping the range will increment the chance of numbers of specific sub range to

111	110	101	100	011	010	001	000	Rule
0	0	0	1	1	1	1	0	30
0	1	0	1	1	0	1	0	90
0	1	1	0	1	0	0	1	105
1	0	0	1	0	1	1	0	150
1	0	1	0	0	1	0	1	165
111	110	101	100	011	010	001	000	Rule
0	0	0	1	1	1	1	0	30
0	1	0	1	1	0	1	0	90
0	1	1	0	1	0	0	1	105
1	0	0	1	0	1	1	0	150
1	0	1	0	0	1	0	1	165

Fig. 2: Transition rules in binary CA

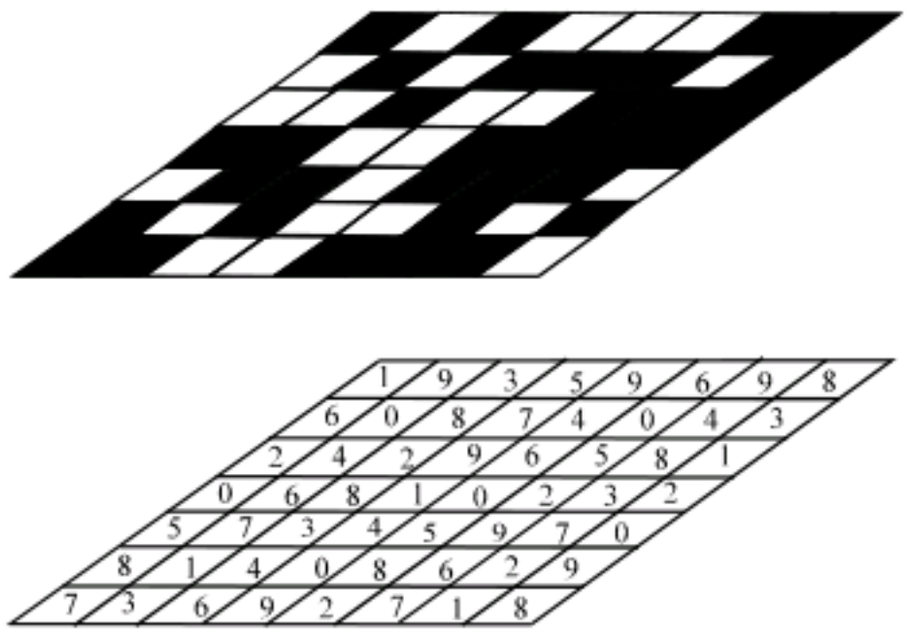

Fig. 3: Two layers CA for uniform random number generation

be generated. For example five bits are needed to generate random numbers in range [0, 20] but five bits envelope range of [0, 31].

The simplest way to handle this problem is to ignore random numbers bigger than 20. However, this method may produce patterns and reduce the quality of generated random numbers. The other approach to compensate this problem is using of linear (or nonlinear) mapping. It is clear that if the length of origin range is bigger than length of destination range, chance of random numbers to be generated won't be the same. In such condition the uniformity will reduce.

Based on this issue, in this study a novel structure of cellular automata is proposed to be used as random number generator. Proposed model is constructed from two heterogeneous layers of cellular automata. Each layer contains a two dimensional cellular automata with same size.

Cells of first layer are binary and include zero or one bits. If the goal is to generate uniform random numbers in range [0, n] then the cells of second layer will include integer numbers between zero and n. structure of proposed model is shown in Fig. 3.

Each row of binary cellular automata-in first layer is assumed as independent linear binary cellular automata.

Thus each cell is adjacent with one right neighbor cell and one left neighbor cell. Associated values of cells of each row will be updated using one of the Wolfram transition rules 30, 90, 105, 110 or 165.

A novel neighborhood structure-named pseudo Neumann-is applied in next layer. In proposed model-like Moore standard neighborhood structure-eight neighbors of each cell are considered as adjacent. The difference between pseudo Neumann and Moore structure is that if cells with same positions from first layer equal one they will be active, otherwise they will be inactive. If the cells of first layer generate uniform random bits, the cells of second layer will activate/inactive with same probability. In this case about half of cells in neighborhood structure-about four cells like Neumann structure-will be active. Interaction between first and second layers of cellular automata leads pseudo Neumann neighborhood structure to be assumed as Neumann neighborhood structure with dynamic adjacency.

States of each cell in second layer will be updated by dividing summation of cell value and active neighbors' values to n+1. Remainder of this division is the next value of cell. According to this rule, values of cells will be between zero and n. Initial configuration of automata must be uniform. In other words chance of integer numbers between 0 to n to be generated must be equal. Now each cell of second layer is a random integer with uniform distribution.

If lower bound of needed random numbers is not zero, it is possible to map the generated random numbers to desired range by using a simple linear mapping. For example if desired range of random numbers is [-100,100] then n will initialized with 200 and -100 will be added to output results.

SIMULATION AND EVALUATION OF PROPOSED MODEL

Here, simulation results of two-layer cellular automata in random number generation will be discussed. Each layer consists of a 1000×1000 cellular automata. States of cells in first layer are updated by using rule 30.

In the following, capability of binary cellular automata to produce random bits by using rule 30 will be discussed and second experiment will compare the uniformity of generated random numbers by proposed model with MATLAB.

Experiment 1: Objective of this experiment is evaluating capability of cellular automata to generate random bits. To reach this purpose, simulation of linear binary cellular automata with one hundred cells was implemented. In this simulation neighborhood radius was one and rule 30 was used to change the cell values 10^3 random bits were generated and total numbers of ones which were appeared in sequence were computed. This simulation had been run for one hundred times and statistical features such as average, standard deviation and scattering length were extracted. Table 1 contains simulation results.

Obviously, Table 1 indicates that quality of generated random bits by using of cellular automata is very desirable and output random bits of automata follow the uniform distribution. Thus if rule 30 is used to update the values of first layer of proposed model, then cells of second layer will activate and inactive with approximately same probability.

Experiment 2: Purpose of this experiment is evaluating the uniformity of generated random numbers in proposed model. Thus, sequence of random numbers is generated by second layer of proposed model and integer random number generator of MATLAB. Then an experiment is implemented as below.

- Generate $N = 10^4$ random numbers in the range of [0,100]
- Classify the generated numbers in c = 10 classes with equal sizes
- Compute the frequency of numbers in each class (f_i)

After running these steps for one hundred times, average, standard deviation and scattering length of frequencies of classes are computed. Table 2 contains the experiment results. Histogram diagrams of calculated frequencies of two-layer cellular automata and MATLAB are shown in Fig. 4 and 5, respectively. Comparing Fig. 4 and 5 results that generated random numbers by two-layer cellular automata are more uniform than MATLAB.

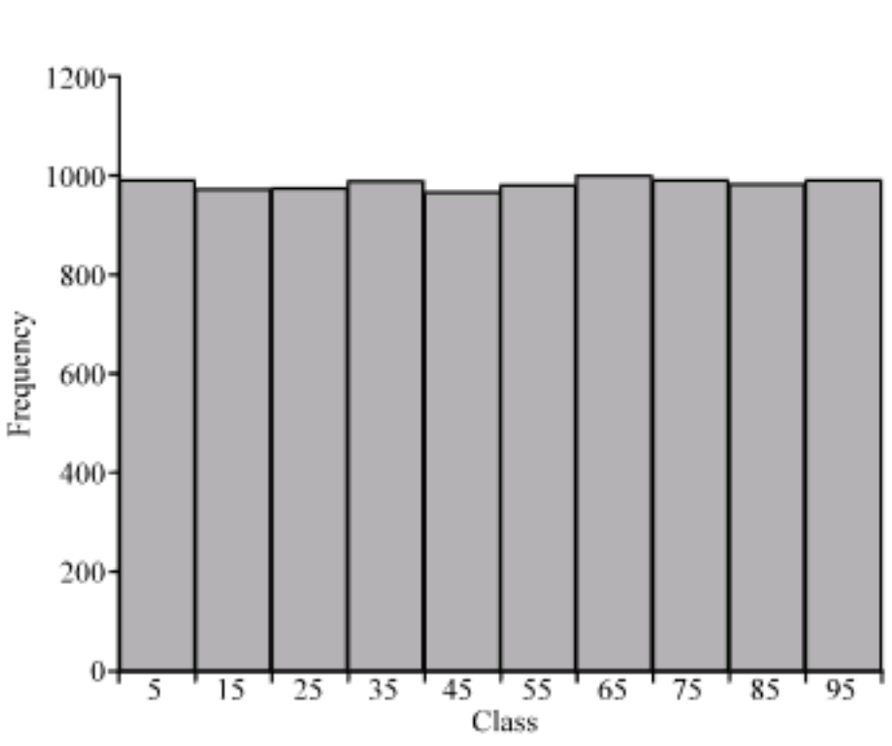

Fig. 4: Histogram diagram of classified generated random numbers by two-layer CA

Table 1: Statistical features of experiment 1

Statistical analysis	Values
Average	499.9484
SD	10.6328
Scattering length	86.0000

Table 2: Statistical features of experiment 2

Method	Average	SD	Scattering length
MATLAB	926.23	111.93	504
MLCA	944.86	83.10	308

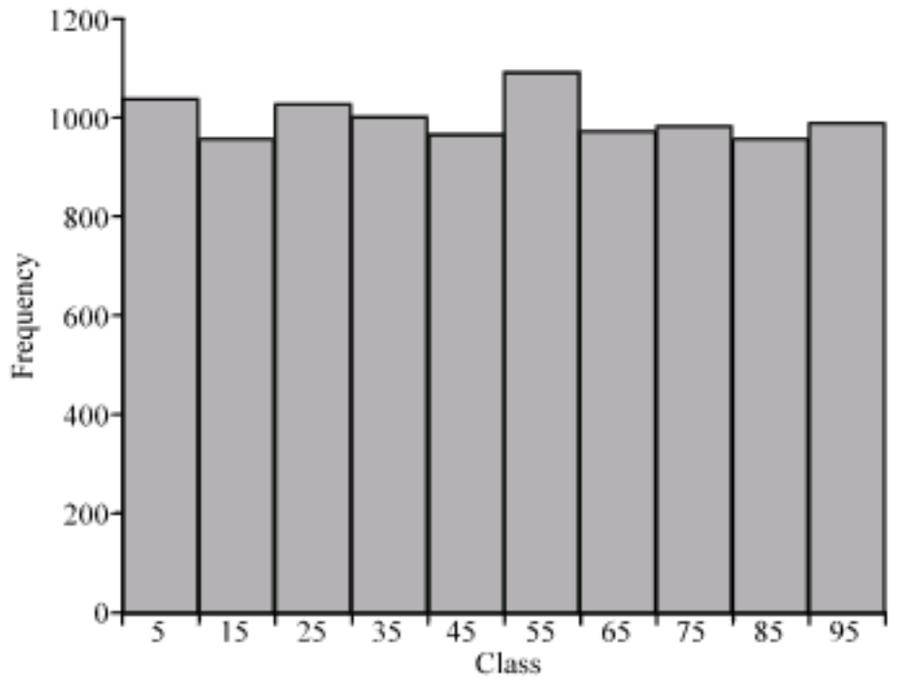

Fig. 5: Histogram diagram of classified generated random numbers by MATLAB

CONCLUSION

In this study, two-layer cellular automata and pseudo Neumann neighborhood structure are introduced. Based on these novel concepts, an innovative approach to generate uniform random numbers is proposed. In this model, cells of automata which are responsible to generate random numbers contain integer values and this is against previous random number generating approaches which were based on generating random bits by using cellular automata. These cells are activated and inactivated by cells of a binary automaton in first layer. This condition leads cellular automata to obtain dynamic neighborhood structure. Simulation results of proposed method and comparing uniformity quality of this method with random number generator of MATLAB proved that two-layer cellular automata generate random numbers with more uniformity. High period, low computational space and time order and low correlation among random numbers are some advantages of proposed model. Also, the output of proposed model can be used as BCD random number. Main problem of this method is initial configuration which may cause Garden of Eden. Applying hybrid transition rules, neighborhood structures and hardware implementation can improve the performance of two-layer cellular automata model.

REFERENCES

Ayanzadeh, R., M. Teshnehlab and S. Setayeshi, 2008. Applying fuzzy operators in cellular automata for uniform random number generation. Proceedings of the 2nd Joint Congress on Fuzzy and Intelligent Systems (9th Conference on Intelligence Systems), Tehran, Iran.

Banks, J., J. Carson, B.L. Nelson and D. Nicol, 2004. Discrete-Event System Simulation. 4th Edn., Prentice Hall, UK.

Bar-Yam, Y., 1997. Dynamics of Complex Systems. Addison Wesley, UK.

Chen, T.Y. and R. Merkel, 2005. Quasi random testing. Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, Nov. 07-11, Long Beach, CA, USA., pp: 309-312.

Hortensius, P.D., R.D. McLeod, W. Pries, D.M. Miller and H.C. Card, 1989. Cellular automata-based pseudorandom number generators for built-in self-test. Proc. IEEE Trans. Comp. Aided Design Integrated Circ. Syst., 8: 842-859.

Jang, J.S.R., C.T. Sun and E. Mizutani, 1997. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. 1st Edn., Prentice Hall, Upper Saddle River, New Jersey, USA., ISBN: 0132610663.

Kalos, M.H., 2007. Monte carlo methods in the physical sciences. Proceedings of the 39th Conference on Winter Simulation, Dec. 9-12, USA., pp: 266-271.

Kohlbrenner, P. and K. Gaj, 2004. An embedded true random number generator for FPGAs. Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, Feb. 22-24, California, USA., pp: 71-78.

Lecuyer, P., 2003. Quasi monte carlo methods for simulation. Proceedings of the 2003 Winter Simulation Conference, Dec. 7-10, Canada, pp: 81-89.

Moghaddas, Y., R. Ayanzadeh and A.T. Hagigat, 2008. A new algorithm for improving the uniformity of random number generators based on calculation with monte carlo method. Proceedings of the Second joint congress on Fuzzy Intelligent Systems (9th Conference on Intelligence Systems), Tehran, Iran, 2008 (In Persian).

Sarkar, P., 2000. A brief history of cellular automata. ACM Comput. Surveys, (CSUR), 32: 80-107.

Viega, J., 2003. Practical random number generation in software. Proceedings of the 19th Annual Computer Security Applications Conference, Dec. 8-12, USA., pp: 129-140.

Wolfram, S., 1986. Cryptography with cellular automata. Proceedings of the Advances in Cryptology, Santa Barbara, California, United States, (CRYPTO'85) Springer-Verlag, New York, USA., pp: 429-432.