International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 9, Issue.7, July 2021 E-ISSN: 2347-2693

Random Number Generator with Long Cycle Based on Memory Cells

Saleh Noman Alassali[1*], Hameed Mansour AL-Aqelee[2]

[1,2]Dept. Computer Engineering, University of Saba Region, Marib, Yemen

[2*]*Corresponding Author: alhomeid66@gmail.com, Tel.: +967-774911720-717091500*

DOI: https://doi.org/10.26438/ijcse/v9i7.3540 | Available online at: www.ijcseonline.org

Received: 16/Jul/2021, Accepted: 20/Jul/2021, Published: 31/Jul/2021

Abstract—Nowadays, random numbers become essential element in many activities. Random numbers can be used in several applications, like simulations, security systems, managements, financial operations, and so on. Indeed, there are many of random numbers generators 'RNGs' currently in used, but so far, there is no ideal one, and the requirement of RNGs are increased while the passed time. The main defects in the available RNGs are the short period of its repeat cycle length and also the predefined values of static factors as well. This research will try to suggest a method to extend the periodic cycle of the repetition, and to improve the quality of the generated numbers randomly. The main idea of this research is to build spherical structure to become memory cells that contain initial random numbers, using a combination of two linear congruently generators. Every cell contains one number. Generating any random number takes place by determining the memory cell index found on spherical structure and make the generation of any random number affects the values of many cells of its neighbors, and also affected by the values of many cells randomly. The spherical structure can be represented by three dimensions matrix, with suitable sizes' not less than 10, 10, 10,.

Keywords—spherical, memory cell, initial values, linear congruently.

I. INTRODUCTION

The importance of RNGs is arisen from the large usability of random numbers in variety of purposes, such as:

a) Generating numbers used as secret code in some finance transactions.
b) Generating data encryption keys.
c) Generating secret numbers used for communication cards
d) Simulating and modelling complex phenomena.
e) Selecting random samples from larger data sets.
f) Test problem generation for the performance evaluation of computer algorithms.
g) Statistical sampling, and so on.

Moreover, nowadays, many of daily activities in security need RNGs to perform their tasks. For example, secret and public-key generation and challenge-response authentication require unpredictable random numbers. Therefore, despite the large amount of theoretical research already done on this subject, many of the generators currently in use, for example the RNGs, are an underlying technology to accomplish highly secure systems. But the main defects in the available RNGs are [1],[2]:

1) The short periods of its repeat cycle length.
2) Sometimes such available RNGs do not satisfied the desire needs for specific applications.
3) The predefined values of static factors may reduce the associated security.

This research will try to suggest a new method to extend the periods of repeat cycle length to generate random numbers, by creating initially appropriate cells contain random numbers and making automatically many changes to the selected sell and it neighbors. The changes will be associated with every generating to any random number. The suggested method can be used for specific application by determine number of digits required. The proposal has been organized in a flexible manner. Section II focuses on the related work. Section III explains a proposal method to generate random numbers. Section IV discusses the analysis & complexity for the proposed method and the experimental result showed in section IV. Conclusion & future research assassinate will be highlighted in section V.

II. RELATED WORK

Indeed, all RNGs are based upon specific mathematical algorithms, which are repeatable and sequential. As such, it would be satisfying to generate random numbers from a process that according to well established understanding. So, to be useful in simulation, a sequence of random numbers R1, R2 ,.., Rn must have at least two important properties: uniformity and independence. That is, each random number Ri is an independent sample drawn from a continuous uniform distribution between 0 and 1(mean 1/2, standard deviation). Some consequences of the uniformity and independence properties are the following [2],[3]:

a) Uniformity: if the interval [0, 1] is divided into n sub-intervals of equal length, the expected number of observations in each interval N/n, where N is the total number of observations. The distribution of numbers in the sequence should be uniform; that is the frequency of occurrence of each of the numbers should be the same.

© 2021, IJCSE All Rights Reserved

b) Independence: the probability of observing a value in a particular interval is independent of the previous values drawn. That means, no one value in the sequence can be inferred from the others. Mainly there are many methods that are used to generate random numbers. In the most methods the modulus (m) should be as large as possible, because a small set of numbers make the outcome easier to predict. Some of them will be mentioned as follows. [2],[3],[4],[5]:

A. Linear Congruently Method (LCM)

This method uses to generate a sequence of integers X1, X2,…Xn between 0 and m-1 by following a recursive relationship as in Eq. (1):

$$Xi = (aXi\text{-}1 + c) \bmod m \quad (1)$$

Such that the specify four parameters are:X0 = seed (or starting value), m =modulus(or divisor), a = multiplier, c = increment; where m > 0 and a < m, c < m, X0< m.

The selection of the values for a, c, m, and X0 drastically affects the statistical properties and the cycle length. The random integers Xi are being generated in the interval [0, m-1]. The random numbers generated, called the stream, are then calculated as in Eq. (2):

$$R_i = \frac{X_i}{m}, \text{ for } i = 1,2,\dots,K. \quad (2)$$

Therefore, the criteria of good RNG is to satisfy maximum density and it leaves no large gaps on [0, m-1]. Thus, to achieve the maximum density and avoid cycling we have to choose the suitable values for the factors a, c, m and X0 as following assumptions [2]:

- For m a power of 2, $m=2^b$ such that b is integer number, a is relative prime to m, and $c \neq 0$. The longest possible period $P=m=2^b$ is achieved if c is relative prime to m and a=1+4k, where k is an integer.
- For m a power of 2, $m=2^b$, and c=0. The longest possible period $P=m/4=2^b$ -2 is achieved if the seed X0 is odd and a=3+8k or a=5+8k, for k=0, 1,...
- For, m a prime and c=0 the longest possible period P=m-1 is achieved if the multiplier a has property that smallest integer k such that ak -1 is divisible by m is k = m-1.

The main drawback in the LCM that, if an opponent knows that LCM is being used and if the parameters are known, and once a single number is discovered, than all subsequent numbers can be easy to know. Even if the opponent knows only that LCM is being used, then the knowledge of a small part of sequence is sufficient to determine the parameters of the algorithm. Suppose that the opponent is able to determine values for X0, X1, X2, and X3. Then

$$X1 = (aX0 + c) \bmod m \quad (3)$$
$$X2 = (aX1 + c) \bmod m \quad (4)$$
$$X3 = (aX2 + c) \bmod m \quad (5)$$

Those equations Eq. (3), (4), (5) can be solved to get the parameters a, c, and m.

B. Combined Linear Congruently Generators

This method obtains the longer period generator because it combines two or more multiplicative congruently generators [2],[6].

- Let Xi,1, Xi,2, …, Xi,k be the ith output from k different multiplicative congruently generators.
- The jth generator X0,j:
- $Xi+1,j = (aj\ Xi + cj) \bmod mj$ (6)
- Such that mj is a prime modulus, aj is multiplier, and mj -1 is a period.
- Produces integers Xi,j approximate ~ Uniform in [0, mj – 1].
- Wi,j = Xi,j - 1 approximate ~ Uniform on integers in [0, mj - 2].

$$X_i = \left(\sum_{j=1}^{k} (-1)^{j-1} X_{i,j} \right) \bmod m_1 - 1 \quad (7)$$

Then the maximum possible period shows in Eq (8) as follows:

$$P = \frac{(m_1 - 1)(m_2 - 1)\dots(m_k - 1)}{2^{k-1}} \quad (8)$$

C. RNG using Cipher Text

This method uses any cipher text to generate random numbers by converting the cipher text to binary digits and selects suitable numbers of binary digits to be converted to decimal digits. But the main defects are that it needs more calculation [3].

III. PROPOSED METHOD

The main idea of the suggested method is to build structure contains sufficient cells with initial values. Two dimensions structure may not sufficient. Figure 1: shows two dimensions structure. So the suggested structure consists of three dimensions, x elements long and y elements wide, and z elements high, Figure 2: shows three dimensions structure. This structure called structure cells SC. SC can be represented by three dimensions matrix, with suitable sizes' not less than 10, like CS[10][10][10], and then generating in the SC suitable random numbers, using a combination of two linear congruently generators. Every cell contains one number. Generating any random number takes place by determining one cell form the structure and making affections on the values of the determined cell and its neighbours.

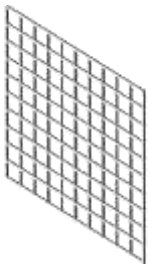

Figure 1: two dimensions

© 2021, IJCSE All Rights Reserved

Every cell in the SC, contains one number with suitable digits. The random numbers in the cells of SC will be used as initial values. Figure 3 shows the structure of the cells in SC without any digits. SC should be dealt as a spherical structure.

Because of dealing with SC as a spherical structure, every cell in SC has 6 neighbours, front, back, right, left, up and down. In case of affecting 9 cells in each layer, the total cells affected are 3*9=27.

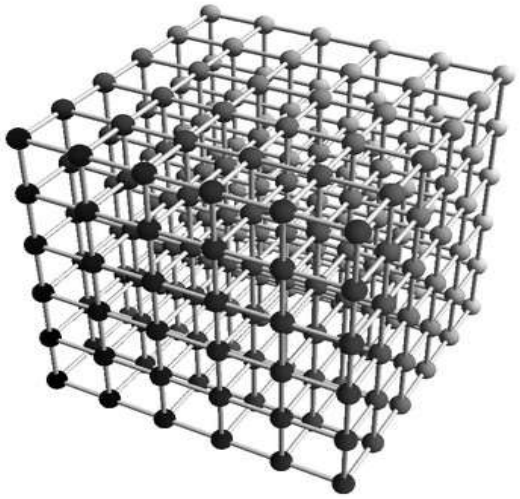

Figure 2 three dimensions structure.

So the following rules should be followed:

1. The neighbors of the right most column in a layer is the left most column in the same layer .
2. The neighbours of a cell in the top most layer, is the cell in the bottom in the same order.
3. The neighbours of a cell in the back side, is the cell in the front side in the same order, finger 2 shows part of the view of SC.
4.

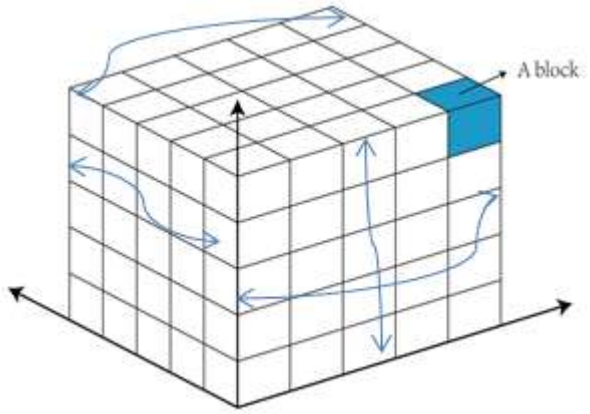

Figure 3: view of SC

Generating random numbers

The seed value is used as index to the three dimensions matrix CS by dividing the seed into four parts as w, x ,y, z . Such that x ,y, z , represent index to the three dimensions matrix and we used to effect the seed value to the next period. The right most, three digits or more, of the seed value are used to represent indexes x, y, and z to specific cell.

The neighbours of the selected cell are specified by computing the lower boundaries neighbours as follows:

```
lx = x-1;   // to get x neighbours
if ( lx< 0)
  lx = lx+ xsize; // to make it circular neighbours.
ly = y-1; // to get y neighbours.
if ( ly< 0)
 ly = ly+ ysize;// to make it circular neighbours
lz = z-1;  // to get z neighbours
if ( lz< 0)
 lz = lz+ zsize; // to make it circular neighbours
```

The higher boundaries neighbours calculated as follows:

```
hx = (x+1 ) mod xsize;
hy = (y+1 ) mod ysize;
hz = (z+1 ) mod zsize;
```

All the values of the cells between the lower boundaries and the higher boundaries will change with every generation of a random number.

The generation for a random number is the new value of the selected cell and some of its neighbours.

The following example shows the indexes of the selected cell and its neighbours, in case of xsize = ysize = zsize= 10.

Example. Let the seed is 2783109. So x=9, y = 0, z= 1, w= 2783. So the selected cell is CS[9][0][1].

The lower boundaries neighbours are lx= 8, ly = 9, lz = 0.

The higher boundaries neighbours are hx =0, hy= 1, hz =2.

The values of the selected cell and its neighbors in the SC will be changed dynamically, simultaneous with every generation to a new random number. Figure 3 shows 27 cells affected with every generation. The values of neighbour cells, are changed depending on the current value of the selected cell. And also every neighbour cell is changed depending on the current value of the selected cell as in equation (9);

$$CS[x][y][z]= (CS[x][y][z]+CS[x_n][y_n][z_n]) \bmod cmaxsize \quad (9)$$

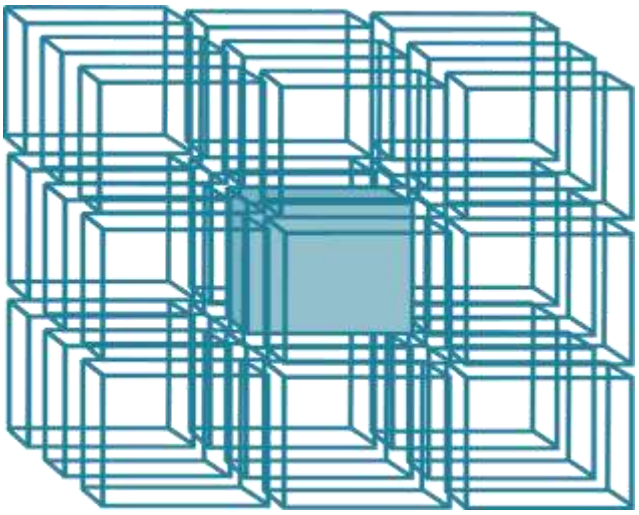

Figure 3

© 2021, IJCSE All Rights Reserved

Such that CS[x][y][z] is the selected cell, CS[x_n][y_n][z_n] is the neighbour cell, and cmaxsize is the maximum value of the cell.

The affection on the selected cell and its neighbours take place by the reminder of addition modulo operation. The reminder of the seed value and the value of the selected cell are used to be the next seed value and so on.

The generation for a new random number will depend on the following factors:

1. The current value of the selected cell.
2. The value of the user seed.
3. The values of the neighbour cells, and the values of the neighbour cells in the upper and lower layers
4. The range of the affected cells from 1 to 3
5. The digits required in each random number.

The additions modulo and or the concatenation of several neighbors cells, in the SC, will be used to compute the required digits as a length of the random numbers.

IV. THE SUGGESTED ALGORITHM

The suggested algorithm becomes as follows:

1. Inputs "seed1, seed2, no, and Nd".
2. Generate initial random numbers based on seed1, save them in the three dimensions array, SC.
3. Compute the index of the selected cell from the seed2.
4. Compute the neighbors of the selected cell
5. Change the values of the selected cell and its neighbors
6. Compute the random number from the selected cell and its neighbors
7. Compute the index of next selected cell
8. Go to 4 or Stop

figure 5 shows the algorithm's operations.

Algorithm discussion

Step 1 input seed1, seed2, no, and Nd. Seed1 is used to generate initial random numbers, that saved in the three dimensions array, called SC. Seed2 is used to compute the index of the selected cell. No is used to determine the required numbers to be generated. Nd is used to determine the required digits in each number.

Step 2: Fill every cell in the SC by generating initial random numbers based on seed1, with suitable digits length. Using the following equations

x0=(7*x+803011)%m2; (10)

x=(x0+c*x)%m1 ; (11)

Step 3: Compute the index of the selected cell current cell using the right most digits of the seed2, using the division modulo operations.

Step 4: Compute the low indexes of the neighbors of the selected cell current cell as follow:

```
void get_Low_Neighbors ( short L[ ], short C [ ] )
{// C[ ] is the current cell indexes
for(short i=0;i<3;i++)
{
 L[i]=C[i]-1; // L[i] is the low neighbors indexes
 if( L[i]<0)
 L[i]=L[i]+size; // size is the volume of SC
 }
return;
}
```

See Table1: SC

Step 5: Change the values of the current cell and its neighbors using addition modulo operations.

Step 6: Compute the new random number, using the values of the current cell and its neighbors.

Step 7: Compute the next current cell, using the values of the last cell of the neighbors.

Step 8: repeat the operations from 4 to 7, or stop if satisfied.

Table 1: shows indexes of some cells in SC

x_{i-1}	y_{j-1}	z_{k-1}	x_i	y_j	z_k	x_{i+1}	y_{j+1}	z_{k+1}
9	9	9	0	0	0	1	1	1
4	9	8	5	0	9	6	1	0
8	0	9	9	1	0	0	2	1
0	9	8	1	0	9	2	1	0
3	7	1	4	8	2	5	9	3
8	8	8	9	9	9	0	0	0

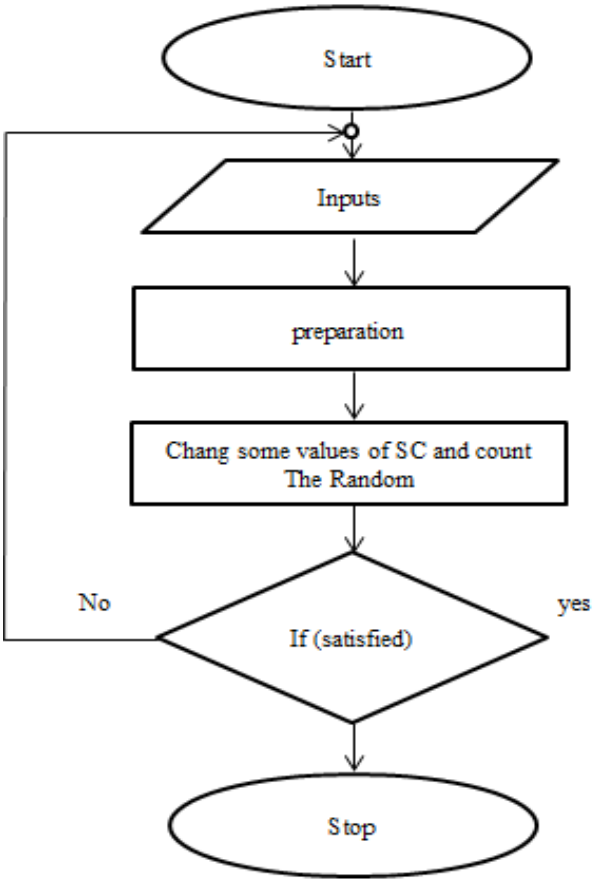

Figure 5:

© 2021, IJCSE All Rights Reserved

Analysis & Complexity Study

Combined linear congruently method, Equations. (10 and 11) are used to generate initial values in CS. Thus the selection of the values for the parameters c, m1, and m2 drastically affect the statistical properties and the cycle length.

The generated number is the result of additions modulo to the values of selected cell and some cells neighbors. Also every cell of the neighbors is affected by the current value of the selected cell. Because of there are 27 neighbour cells, the quality of the generated number improved, also the possibility of getting sufficient digits are available using concatenation and or addition of more neighbour cells.

The circulation of the generated number may occur after huge numbers of generations.

The disruption of the digits appear in table 2 when generating 100 numbers, the average digits is 20,5 in each number.

Table 2:

Digits	Repetitions
0	183
1	202
2	189
3	194
4	193
5	210
6	188
7	219
8	193
9	196

Table 3:

Digits	Repetitions
0	582
1	696
2	694
3	686
4	699
5	732
6	691
7	676
8	673
9	685

Table 3 shows the disruption of the digits when generating 300 numbers, the average digits is 24,5 in each number.

The security of the algorithm depends upon two seeds numbers and three values act as a factors generations in the algorithm.

V. EXPERIMENTAL RESULTS

This section shows samples of experimental results using own generated simulation software according to the proposed algorithm mentioned in section III. The result shows 100 numbers, with average 20,7 digits. The result shows that, there is no reputation values and the cycle period is better. Table 2&3 show the disruption of the digits when generating 100 and 300 numbers.

49892190563795105873 8 6964594105137274176
72753506407654957154 97 51708089309430403836 0
89175080602650314937 49 25304021267856968279
23450309781892939293 2 96317953414057468453 5
57605826820482489171 6 40917928019652517460 1
19654927317141646469 97674704563173282843 8
63760808084191038206 4 76517515301390719214 8
63072192427981825868 27646342474072460285 1
26017898364932451525 49735391717404697150
71329090380468590517 0 57401736131703708454 6
23915247841635234328 1 91769106594359859315 3
95367618472851919521 3 61954181091957678691 4
26292592769456520647 8 36853952101794895365
92423239396781710782 6 95962768509859320203 6
74537284706813294347 0 56703685160296476932 7
46906072619073170143 1 19520952073956319193 8
74974614359532079450 2 84360481517139057053 4
48683950917919086806 5 38209650627978464172
17329782496763681476 9 62354602902794857161
39284836170157650215 51565814247205291915 7
76780383027465941287 5 49476362587803070703
46735303508308473082 18534071795807405989 1
73527048420484248720 8 14856725686846272527 3
13684167421974028569 39187202120353818175
54831654210625109873 8 06432307621971783625 9
75930506312751208092

IV. CONCLUSION AND FUTURE WORK

Any random number generator required seed number and some factors. It is difficult to detect the factors used to generate random numbers if the seeds change periodically. Also it is more difficult to expect the next generated random number based on the idea of combined linear congruently generators with three parameters, like a_i, c_i, and m_i. Using structure memory cells, generate random numbers based on dynamic seeds and dynamic changes to the selected cell and its neighbours make the cycle period longer, and make the difficulty is more to expect the next generated random number from the previous one. The result shows that, there is no repetitions values and the cycle period is better than the other algorithms. In the future-work, the method needs software application to mix two or three numbers as one.

REFERENCES

[1] William Stallings, "Cryptography and Network Security: Principles and Practice" 3rd Ed. India Reprint. Agrawal-M IETE-Technical-Review. **2009.**

[2] Jerry Banks, etc.., "Discrete-Event System Simulation", 3th Ed. Pearson Education, Singapore. **2001.**

[3] Bruce Schneier, "Applied Cryptography" 3rd Edition John Wiley & Sons. (ASIA) Pvt. Ltd., 2 Clementi Loop # 02-01, Singapore 129809. **2010.**

[4] Borosh. S., and Niederreiter, H., "Optimal Multipliers For Pseudo-Random Number Generation By The Linear Congruential Method", BIT 23,65-74. **1983.**

[5] Figiel, K.D., and Sule. D.R., "New Lagged Product Test for Random Number Generators". Comput. Ind. Eng. **Vol. 9, 287-296, Mar. 1985.**

[6] P. L'Ecuyer, "Efficient and portable combined random number generators", Communications of the ACM 31 **Volume 31 Number 6, USA, June 1988**

© 2021, IJCSE All Rights Reserved

AUTHORS PROFILE

Mr. **Hameed Mansour Abdo Sedd Al Akelee**, received a Bachelor of Information Technology from Al Hoduda University, Yemen. He is currently pursuing Master of Information Technology , Department of Computer Engineering of Alandalus University, Sana,a Yemen. His areas of interest are Information Security, C# Programming.

Dr. Saleh Noman Abdullah Alassali, he got Bachelor of computer engineering from KSU University, soudia arabia in 1988, he got master degree in Computer Science from Pune University, India in 2000, he got the Ph.D. in Infromation Securty, SRTMU, India, in 2005. In 2007 he became the head department of Computer Science Marib College, Sana'a University, Yemen. Currently he works as Associate Prof. in Computer Science department Sheba University, Marib, Yemen. He has published more than 8 reseach papers in reputed international journals. His main research work focuses on Cryptography algorithms and random number generators.
WhatsAPP: 770317665, Yemen.

© 2021, IJCSE All Rights Reserved