ISSN:0975-9646

Cellular-Genetic Key Generation Algorithm

Neha Singla

Assistant Professor, Department of Computer Science and Engineering
Manav Rachna College of Engineering,
Faridabad, India

Abstract— **The Genetic Algorithm (GA) requires randomized initial population so as to give requisite results. The work proposes the use of Cellular Automata (CA) to generate the initial population for GA. The numbers generated by clubbing these two processes will make a better set of random numbers as compared to existing methodologies. The quality of the random numbers generated has been tested by Gap Test and Karl Pearson coefficient of correlation test. So as to ensure the goodness of random numbers coefficient of auto correlation has also been calculated. This technique can be used to generate keys in cryptography and it will be nearly impossible to guess the key as both the techniques are of natural ethos and not mathematical in nature.**

Keywords— **Random Number Generator, Cellular Automata, Genetic Algorithms, Cryptography, Vernam Cipher.**

I. Introduction

Genetic Algorithms (GAs) are a type of optimization algorithms which combine survival of the fittest and a simplified version of genetic process [1]. Cellular Automata are distributed system which can perform complex computation. It has as yet not been established whether cryptography can be considered as a problem apt for applying the combination of CA and GAs. Therefore the work explores the use of combination of CA and GAs in cryptography. A detailed study on the success of GAs in cryptography was carried out by Bethany Delman [2] but it was limited to breaking the cipher text with GAs. This work takes the example of Vernam Cipher to apply both CA and GAs and proposes a new technique to produce a key which can substitute One Time Pad (OTP) or Pseudo Random Number Generator (PRNG) [2].

The patterns of cellular automata are predictable and well defined. The problem therefore was reduced to change them to unpredictable patterns. The analysis carried out suggested that few of the 256 patters; 48 to be precise; are good contenders of initial population generation module of GAs. The patterns generated were subjected to a technique described in the following sections to get the initial set of chromosomes. These were put to crossover and mutation operation to get the final population. The complete technique has been implemented and the randomness of the population generated calculated. The experiments carried out established the ability of CA and GAs to produce a good random sample. If the key of the Vernam Cipher is selected from that sample then it is found to be less predictable as compared to PRNG used by Microsoft Visual C#.

II. Random Numbers

Random number generation is the art of producing pure gibberish as quickly as possible. According to Eric Hoffer "creativity is the ability to introduce order into the randomness of nature". So in order to be creative also we need random numbers. It has been shown that most of the random number generators cannot be regarded as a 'true' random number generator. Since its output is predictable [4]. The physical method of producing random number may include atomic or subatomic physical phenomenon. The need to generate random number as early as possible for cryptographic systems led to the creation of random bit generator Working at 300 gigabit per second at the bar LLAN University in Israel .In a random number generator even the slightest pattern cannot be tolerated. To detect such bias, we have a wide variety of tests. Test results are usually reported as autocorrelation measure. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero. In our case the autocorrelation measure is approximately 0.04, which is considered as a good random sample. Moreover the above method opens window of AI to a new process of generation of Random numbers.

III. Cellular Automata

The history of CA dates back to the forties. It was started by Stanislaw Ulam. His aim was to study the evolution of graphic constructions generated by simple rules. The base of the construction was a two-dimensional space divided into cells. Each of these cells had two states: ON or OFF. Starting from a given pattern, the new generations were formed according to neighborhood rules. Ulam found that this mechanism permitted to generate complex figures and that these figures could be self-reproducing. Simple rules were made to build very complex patterns. The question that needs to be analyzed is whether the complexity is apparent or real [5]. These rules were studied by John von Neumann for use in self-reproductive automata .He worked on the conception of a self-reproductive machine, the "kinematic". Such a machine was supposed to be able to reproduce any machine described in its programs; including a copy of itself. This led to libration from real physical constraints to work in an extremely simplified universe that was able to generate a high complexity. The use of this formal universe led him to notice:

"By axiomatizing [self-reproductive automata with cellular automata], one has thrown half of the problem out the window, and it may be the more important half. One has resigned oneself not to explain how these parts are made up of real things, specifically, how these parts are made up of actual elementary". Van designed 29 states cells, containing a universal replicator, a description of itself and a Turing machine for supervision. CA left laboratories in 1970 with the now famous Game of Life of John Horton Conway.

IV. Genetic Algorithms

A genetic algorithm is a search heuristic that mimics the process of natural evolution used to generate useful solutions to optimization and search problems. Genetic Algorithms are a subset of what we call evolutionary algorithm which solves optimization problem using techniques inspired by natural evolution such as inheritance, mutation, selection, and crossover [6].

John Holland from the University of Michigan started his work on genetic algorithms at the beginning of 60s. A first achievement was the publication of Adaption in Natural and Artificial System in 1975. Holland has two aims, first to improve the understanding of natural adaption process, second to design artificial systems having properties similar to natural systems. Holland method considers the role of mutation and also utilizes genetic recombination that is crossover to find the optimum solution.

Crossover and mutation are two basic operators of GA. Performance of GA depend on them. Type and implementation of operators depends on encoding and also on a problem.

There are many ways of how to do crossover and mutation.

A. *Crossover*

1) Single point crossover: In this case one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, and the rest is copied from the second parent.

11001011+11011111 = 11001111

2) Two point crossover: Here two crossover point are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the second parent and the rest is copied from the first parent.

11001011 + 11011111 = 11011111

3) Uniform crossover: In this method bits are randomly copied from the first or from the second parent.

11001011 + 11010101 = 11010101

B. *Mutation*

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of algorithm chromosomes to the next. It is similar to biological mutation.

Method given in most of the sources including Wikipedia: An example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are inversion and floating point mutation. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions and scrambles.

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other.

C. *Reproduction and Selection*

Chromosomes are selected from the population to be parents to crossover. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection.

1) Roulette Wheel Selection: Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where are placed all chromosomes in the population, every chromosome has its place big accordingly to its fitness function.

V. Proposed Work

The following process uses CA to generate the initial population of GAs and implement a Random Number Generator (RNG) which uses the best parts of both the techniques. The RNG generated by the method given below retains the simplicity of the genetic process and is more efficient then the key generation processes of DES and AES. Still it gives result comparable to both of them. The process has been explained in this section

A. *Generate Elementary Cellular Automata*

Elementary Cellular Automata is produced for all the 256 rules in the form of bits (0 and 1).

256 matrices of 100 X 100 are obtained.

B. *Matrix Reduction*

For each matrix rows are divided into 10 equal parts, resulting in 10 bits in each part.

For each part majority is to be calculated. If number of one's is greater than or equal to number of zeroes, then, that part is replaced by 1; else it is replaced by 0.

Thus, 100 X 100 matrix is reduced to 100 X 10 matrix.

Thus, 256 matrices of 100 X 10 are obtained.

C. *Analysis*

Matrices resulting in random patterns are considered and analyzed and 48 such matrices are separated. These matrices will be used in the following steps to produce the initial population of Genetic Algorithm.

D. *Random Selection of Rule*

A rule is randomly selected from a set of 48 values and its reduced matrix is obtained.

E. *Crossover*

From the above reduced matrix of 100 X 10, 2 random rows are selected and Crossover operator is applied on them and a new row is obtained.

F. *Mutation*

Mutation Operator is applied on the above row and new random population is obtained.

G. *Decimal Representation*

Decimal representation of above obtained row is calculated, and can be used as a key.

Autocorrelation test, Gap test and Karl Pearson coefficient of correlation test is applied on a data of keys obtained.

Fig. 1 explains the above process.

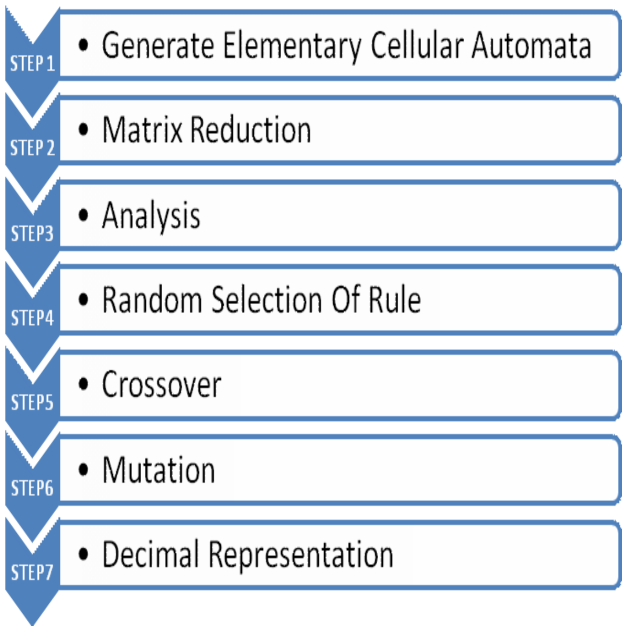

Fig. 1 Example of an image with acceptable resolution

Due to limited resources and time, decimal representation has been used in step 7. Instead of operating the row obtained after mutation operator with 2, a random number between 2 to 100 can be generated and applied on the population obtained in step 6, thus increasing the range of random numbers. By this variation the range of random number become so large that the given technique, if carefully analyzed and accepted will have the capability of surpassing mathematical algorithms like AES.

VI. Conclusions

The above technique has been implemented and samples have been generated. The implementation has been done in C#. Samples have been collected and analyzed in Microsoft Excel, Various tests have been applied on the sample and most of them give satisfactory results.

Since, around a 1000 values were analyzed and no repetition was obtained therefore frequency test was not applied. The coefficient of autocorrelation was calculated for k = 1 to k = 19. The result for k = 1 was 0.04, thus indicating a good random sample.

In a next test, a time series has also been considered and Karl Pearson Coefficient of correlation has been calculated, also giving satisfactory data.

In the analysis of data, a correlogram is an image of correlation statistics. In time series analysis, a correlogram, also known as an autocorrelation plot, is a plot of the sample autocorrelations r_h versus h (the time lags). The correlogram is a commonly used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero. Such correlogram has also been plotted for our sample data.

The majority was calculated by taking a group of 10 cells. If more samples are needed then a set of 5 cells can also be taken. The whole process is being enhanced and analyzed.

References

[1] H. Bhasin, and N. Arora, "Key Generation for cryptography using Genetic Algorithms," *proceedings of (ICRITO) International Conference on Reliability, Infocom Technology and Optimization*, pp 226 – 231, 1 – 3 November, 2010.

[2] B. Delman, "Genetic Algorithms in cryptography," Published in web, July 2004.

[3] G. S. Verman, "Cipher printing Telegraph Systems for secret wire and radio Telegraph Communications", *Journal of IEEE*, vol 55, pp 109 – 115, 1926.

[4] H. Bhasin, "Corpuscular Random Number Generator," *proceedings of International Conference on Network Communication and Computer 2011, IEEE*, pp 541 – 543, March 21 – 23, 2011.

[5] S. Wolfram, "Cellular Automata and Complexity: Collected Papers," ISBN 0-201-62716-7, 1994.

[6] H. Bhasin and S. Bhatia, "Application of Genetic Algorithms in Machine learning", *(IJCSIT) International Journal of Computer Science and Information Technologies*, Vol. 2 (5), pp 2412 – 2415, 2011.

[7] H. Bhasin, S. Arora, "Use of Genetic Algorithms for finding roots of algebraic equations", IJCSIT, Vol. 2(4), pp-693-696.

[8] H. Bhasin, N. Singla, "Cellular automata based test data generation", ACM Sigsoft software engineering notes, Vol. 38(4), pp- 1-7.

[9] H. Bhasin et. al. " regression testing using fuzzy logic", IJCSIT, Vol. 4(2), pp 378-380.

[10] H. Bhasin, N. Singla, "Genetic based algorithm for N-puzzle problem", IJCA, 51(22), pp-44-50.

[11] Harsh Bhasin et. al," Harnessing Genetic Algorithm for vertex cover problem", IJCSE, 4(2).