Information Technology Journal 7 (3): 541-544, 2008
ISSN 1812-5638


# Finding Linear Equivalence of Keystream Generators Using Genetic Simulated Annealing

Wasan Shaker Awad
Department of Information Systems, College of Information Technology,
University of Bahrain, P.O. Box 32038, Sakhair, Bahrain

**Abstract:** In this study, we propose a new approach for finding the linear equivalence of key stream generators. This study presents two simulated annealing algorithms which have been applied to solve this problem. Both algorithms use genetic operations to modify the candidate solutions. The performance evaluation is carried out for a set of key stream generators and the results are compared with the results of applying genetic algorithm to solve the same problem. The proposed algorithms are able to find the linear equivalence and more effective than genetic algorithm.

**Key words:** Simulated annealing, genetic algorithm, stream cipher, linear equivalence

## INTRODUCTION

Stream ciphers play important role in cryptographic practices and generated pseudorandom sequences with good properties are essential components in a wide variety of modern applications such as radar. Stream cipher is a different class from block cipher that can operate one data unit as small as a bit or a character (Schneier, 1996). Every stream cryptosystem consist of two parts, which are:

- Key stream generator and
- Mixer (XOR for the binary sequence).

A key stream generator outputs a stream of bits (key stream), which is xored with a stream of plaintext bits to produce the stream of cipher text. Mainly, stream cipher systems can be classified into linear and nonlinear stream ciphers. In the linear ciphers, the key stream generators are linear feedback shift registers (LFSRs) which are shift register with linear feedback functions called characteristics polynomials, while, in nonlinear ciphers, the key stream generators are composed of a number of LFSRs combined using a nonlinear combination function, or composed of a shift register with a nonlinear feedback function, which is called nonlinear filter.

The stream cipher system's security depends entirely on the inside of key stram generator. The security of this generator can be analyzed in terms of randomness, linear complexity and correlation immunity. Thus, good stream cryptosystems must have the following features (Zeng *et al*., 1991).

- They generate long period key streams from short keys.
- Their key streams are random and of large linear complexity.
- They have high degrees of correlation immunity.

Linear complexity is a well-known complexity measure in the theory of stream ciphers, which is first studied by Massey (1969). Linear complexity of a key stream s is the length of the shortest LFSR which will produce the stream s, which is denoted by L(s). If the value of L(s) is L, then 2L consecutive bits can be used to reconstruct the whole sequence. Hence, to avoid the key stream reconstruction, the value of L should be large.

Linear complexity plays a crucial role in designing good stream cipher systems. Therefore, it has been studied by various authors; some researchers concentrate on finding upper or lower bounds on the linear complexity such as (Caballero-Gil, 2000). Garcia and Fuster-Sabater (2000) concentrate on analyzing the linear complexity of nonlinear filter key stream generators, who mentioned in their paper that there is no efficient method to determine the value of linear complexity of the sequence generated by nonlinear filtering applied to LFSR.

This study is to present a new approach for solving the problem of finding the linear equivalence. Our goal is to find an effective method that can be used to find the linear equivalence and hence the value of linear complexity, for any binary sequence. Therefore, three algorithms have been developed which are based on Simulated Annealing (SA) and Genetic Algorithm (GA). This study includes the description of the proposed

algorithms, study of the effectiveness and performance of these algorithms, in addition to a comparison between SA and GA.

## SIMULATED ANNEALING AND GENETIC ALGORITHMS

Kirkpatrik *et al.* (1983) introduced SA as a global optimization heuristic that is inspired by the annealing process in metallurgy. At high temperatures, the molecules of liquid move freely with respect to one another. If the liquid is cooled slowly, thermal mobility is lost. SA is a general randomization technique for solving optimization problems. This technique can help to avoid the problem of getting stuck in a local minimum and to lead towards the globally optimum solution (Yong *et al.*, 1995). In SA, the solution starts with a high temperature and a sequence of trail vectors are generated until inner thermal equilibrium is reached. Once the thermal equilibrium is reached at a particular temperature, the temperature is reduced and a new sequence of moves will start. This process is continued until a sufficiently low temperature is reached, at which no further improvement in the objective function can be achieved (Cheng *et al.*, 2007). Thus, SA algorithm consists of: configurations, re-configuration technique, cost function and cooling schedule.

Many researchers explored the application of SA on many different types of problems, such as cryptographic problems. It has been used, for example, to design S-boxes and Boolean functions (Clark *et al.*, 2004, 2005; Clark and Jacob, 2004) and attacking cipher systems (Golic *et al.*, 1996).

SA can be integrated with GA in order to work on a population of individuals and to preserve good individuals into the next generation (Yong *et al.*, 1995; Sadegheih, 2007; Wong *et al.*, 2005). GA (Holland, 1975; Goldberg, 1989) is a general, probabilistic and adaptive search algorithm. GA is a stochastic search algorithm that is based on the Darwinian principle of survival of the fittest. It works on population of individuals (chromosomes) that represent the candidate solutions. GA can be summarized as follows:

- Determine a representation scheme for the candidate solutions.
- Generate the initial population randomly.
- Evaluate the fitness for each chromosome.
- Select members of the population for crossover and mutation to produce a population of children.
- Repeat steps (3-4) until stopping criteria is satisfied.

## GENETIC METHOD

First, we are going to describe GA used to solve the problem under study. The following is the description of GA for finding the linear equivalence.

**Representation scheme:** In the GA method, variable length chromosomes are used. The binary representation has been chosen to represent the chromosomes that represent the characteristic polynomials of LFSRs. These chromosomes are the candidate linear equivalences for a given key stream of length (lenkey) which is supposed to be 2 L. The minimum chromosome length is 2 bits and the maximum length is lenkey bits.

**Fitness function:** Fitness function is used to evaluate each chromosome c. In the proposed method, the fitness function is given by Eq. 1.

$$Fit(c) = (lenkey-er) + (lenkey/3)*(1.0/(1.0 + l(c))) \quad (1)$$

The fitness value of each chromosome c is assigned by applying the following algorithm:

**Algorithm fitness value calculation:**

Input: The first lenkey bits of the Keystream and c.
Output: Fitness value of c.
Begin
Let l(c) be the length of the chromosome c to be evaluated;
Let w be the initial state of LFSR, which is the first l(c) bits of the keystream;
Use c and w to generate a binary sequence g(c) of length lenkey;
Calculate the difference between the keystream and g (c), i.e., error (er);
Compute Fit(c);
End;

The fitness function includes two terms, the first term (lenkey-er) is to evaluate the correctness of the generated key stream and hence the LFSR. The second term is to evaluate the optimal solution, which is the shortest LFSR. This measure is weighted by (lenkey/3) which has been chosen since it gives the best result.

**GA parameter:** The genetic operation used to update the population is 1-point crossover and mutation with probability 5%. The selection strategy, used to select chromosomes for the genetic operations, is the 2-tournament selection. The population size (n) is

dependent on the length of known key stream (lenkey). The population size n increases as lenkey is increased. Therefore, the population size is computed using Eq. 2.

$$n = lenkey * 10 \quad (2)$$

The old population is completely replaced by the new population which is generated from the old population by applying the genetic operations. The worst chromosome in the new population is replaced by the best chromosome of the old population.

**Stopping criteria:** The run of GA is stopped after a fixed number of generations. The solution is the best chromosome in the last generation. The solution should be of fitness value (Lenkey + L), which indicates that the generated key stream is free of error and the chromosome represents the shortest LFSR, i.e., linear equivalence.

## GENETIC SIMULATED ANNEALING METHODS

This part of research is to describe the proposed genetic annealing algorithms (GSA1) and (GSA2) for finding the linear equivalence. The representation scheme of the chromosomes, genetic parameters and the fitness functions of genetic method are also used in GSA1 and GSA2. The following are the algorithms. You can see the difference between the two algorithms; the first one is population-oriented and the second one is chromosome-oriented.

**Algorithm GSA1:**

```
Generate the initial population (pop) and evaluate it.
Temp = 250.0; //temperature
  Repeat
    Generate a new population by applying crossover and mutation, (pop1);
    Evaluate the fitness of the new generated chromosomes of pop1;
    Calculate the averages of fitness values for pop and pop1, av1 and av2;
    If (av2 > av1) then replace the old population by the new one, i.e. pop = pop1;
    Else
      Begin
        e = av1-av2;
        Pr = e/Temp;
        Generate a random number (rnd);
        If (exp(-pr) > rnd) then pop = pop1;
      End;
    Temp = temp * 0.99;
  Until (Max Number of generations);
```

**Algorithm GSA2:**

```
Generate the initial population (pop) and evaluate it.
Temp = 250.0; //temperature
  Repeat
    Repeat n/2 times //population size
    Begin
      Select two chromosomes (parent) from the current population;
      Generate two chromosomes (children) using crossover and mutation;
      Evaluate the fitness of the children;
      Calculate the averages of fitness values for parent and children, av1 and av2;
      If (av2 > av1) then copy the children to the new population;
      Else
        Begin
          e = av1-av2;Pr = e/Temp;
          Generate a random number (rnd);
          If (exp(-pr) > rnd) then copy the children to the new population;
          Else copy the parent chromosomes;
        End;
    End;
    Temp = temp * 0.95;
  Until (Max Number of generations);
```

## EXPERIMENTAL RESULTS

The main purpose of this paper is to study the effectiveness of GSA to solve the problem of finding the linear equivalence of key stream generators. Therefore, the proposed methods have been implemented using C++ programming language and eleven different key streams of different linear complexities, from 3 to 13, have been used as input. These key streams have been generated by LFSRs of characteristics polynomials shown in Table 1. The maximum number of generations used is 30 generations and the input key stream length is 2L bits.

Table 1: LFSRs used in the experiments

| The characteristic polynomial used | L |
|---|---|
| 1101 | 3 |
| 11001 | 4 |
| 111011 | 5 |
| 1000011 | 6 |
| 10111111 | 7 |
| 100101101 | 8 |
| 1100011111 | 9 |
| 10100110001 | 10 |
| 111011111001 | 11 |
| 1010111101011 | 12 |
| 11010111101001 | 13 |

Table 2: The probability of algorithms' success (P)

| LP (%) | GA | GSA1 | GSA2 |
|---|---|---|---|
| 3 | 100 | 100 | 100 |
| 4 | 86 | 99 | 100 |
| 5 | 92 | 99 | 89 |
| 6 | 56 | 94 | 61 |
| 7 | 57 | 90 | 70 |
| 8 | 23 | 50 | 37 |
| 9 | 15 | 23 | 33 |
| 10 | 8 | 18 | 27 |
| 11 | 4 | 21 | 17 |
| 12 | 2 | 10 | 15 |
| 13 | 3 | 5 | 12 |

Table 2 shows the probability of success (P). P is calculated by executing the proposed methods 100 times for each key stream (100 trails) and the method success to find the solution if the best chromosome of the last generation is the desired LFSR, i.e., shortest LFSR that generates the given key stream.

First of all, we can see that P decreases as L increases using all methods. This result was expected, that is because, as L increases longer key stream (of length 2L) is required to be tested by the fitness function. But this result may be improved by increasing the number of maximum generations or population size. Furthermore, the integration of SA and GA has improved the performance, such that, the probability of success of GSA1 and GSA2 is greater than of GA and GSA2 gives better results for large linear complexities comparable with GSA1.

## CONCLUSION

New methods for finding the linear equivalence (and the value of linear complexity) for any Binary key stream knowing the first 2L bits have been developed. These methods are based on interesting intelligent techniques which are GA and SA. The results show that GA and SA are capable of solving the underlying problem, but an integration of SA and GA gives better results, that is, the probability of success is increased by using SA.

## REFERENCES

Caballero-Gil, P., 2000. New upper bounds on the linear complexity. Comput. Math. Appl., 39 (3): 31-38.

Cheng, Y.M. *et al*., 2007. Performance studies on six heuristic global optimization methods in the location of critical slip surface. Doi:10.1016/j.compgeo.01.004.

Clark, J.A. *et al*., 2004. Searching for Cost Functions. Evolutionary Computation, CEC20004 Special Session on Security and Cryptology, Congress, Portland Oregon, USA., pp: 1517-1524.

Clark, J.A. and J.L. Jacob, 2004. Almost Boolean functions: The design of Boolean functions by special inversion. Comput. Intel., 20 (3): 450-462.

Clark, J.A. *et al*., 2005. The design of S-boxes by simulated annealing. Evolutionary Comput., 23 (3): 219-231.

Garcia, L.J. and A. Fuster-Sabater, 2000. On the linear complexity of the sequences generated by nonlinear filtering. Inform. Process. Lett., 76 (1): 67-73.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

Golic, J.D.J. *et al*., 1996. Discrete optimization and fast correlation attacks. Lecture Notes Comput. Sci., 1029: 186-200.

Holland, J.H., 1975. Adaptive in natural and artificial systems. Ann Arbor, University of Michigan.

Kirkpatrik, S. *et al*., 1983. Optimization by simulated annealing. Science, 220 (4598): 671-680.

Massey, J.L., 1969. Shift register synthesis and BCH decoding. IEEE Trans. Inform. Theor. IT., 15 (1): 122-127.

Sadegheih, A., 2007. Sequence optimization and design of allocation using GA and SA. Applied Math. Comput., 186 (2): 1723-1730.

Schneier, B., 1996. Applied Cryptography. John Wiley and Sons.

Wong, Z., M. Rahman and Y. Wong, 2005. GECCO' 05, June 25-29, Washington, DC, USA.

Yong, L., K. Lishan and D.J. Evans, 1995. The annealing evolution algorithm as function optimizer. Parallel Comput., 21 (3): 389-400.

Zeng, K., C. Yang and T.R.N. Rao, 1991. Pseudorandom bit generator in stream cipher cryptography. Computer, 24 (2): 8-17.