Recovering the Cyclic-Code of Generated Polynomial by Using Evolutionary Computation

Kangshun Li [1 2 3], Yuanxiang Li [1 2], Haifang Mo [2]

[1] State Key Laboratory of Software Engineering, Wuhan University, Wuhan, 430072, China.
[2] Computer School of Wuhan University, Wuhan, 430072, China
[3] School of Information Engineering, Jiangxi University of Science & Technology, Jiangxi, 341000, China.
Li Kangshun,lks@public1.gzptt.jx.cn

Abstract

The data integrity in computer security is a key component of what we call trustworthy computing, and one of the most important issues in data integrity is to detect and correct error codes, which is also a crucial step in software and hardware design. Numerous methods have been recently proposed to solve legal-codes of the cyclic-code generated polynomial g(x). We think that a better approach for this purpose is to solve the legal-codes by finding the roots of the cyclic-code generated polynomial. However, as it is well known, finding roots of polynomials of high degree in the modulo-q space GF(q) is very difficult. In this paper we propose a method to solve the roots of cyclic-code generated polynomial by using evolutionary computation, which makes use of randomized searching method from biological natural selection and natural genetic system.

1 Introduction

With the fast development of computer, there are more and more demand on computer application in all fields, and the trustworthy computing play more and more important role in the software design and network communication. The computer detecting error codes and correcting error codes is the basic demand of trustworthy computing, therefore, the computer detecting error codes and correcting error codes is of significance to trustworthy computing and to computer security. The precise bits of detecting error codes and correcting error codes[1] are 64 bits nowadays in MILKWAY- II computer while there was only 1 bit in the past, and in the future, the spaceflight technology of shipping persons and GPS will all request higher capacity of error-correcting and error-detecting. Therefore, we need more scientific and more complete trusted computing means. This paper introduces evolutionary algorithm with simulative biologic genetic evolution through population crossover and mutation to attain the solutions of problem, and to solve the roots of exponential cyclic-code generated polynomial $g(x)$ in modulo-q space $GF(q)$.

2 The Theory of Evolutionary Computation

The method of evolutionary computation[2] is a random searching method referencing biological natural selection and natural genetic mechanism. It includes genetic algorithms (GAs), evolution strategies (ESs), evolutionary programming (EP) and genetic programming (GP). It is also a new method to solve problems through combining nature genetics with computer science.

If we consider an optimizing problem as following:

$$\min\{f(x) \mid x \in X\}$$

Foundation item: This work was supported by the National Natural Science Key Foundation of China with the Grant No.60133010 and the National Research Foundation for the Doctoral Program of Higher Education of China with the Grant No.20030486049.

Biography: Li Kangshun (1962-), male, Ph. D candidate, senior engineer, research direction: Evolutionary computation, computer security.

subject to

$$g_i(x) = 0, \quad i = 1,2,\cdots,n$$

where $f(x)$ is an objective function in domain X, $\forall x \in X, f(x) \geq 0$ 。 X is a feasible set of its solutions. It can be a finite set (for example, a combination optimization problem), it can also be a subset of real space R^n (for example, continuous optimization problem) etc.

3 Steps of Evolutionary Computation Algorithm

Step 1: Producing the initial population $P(t) = \{x_1(t), x_2(t), \ldots\ldots, x_n(t)\}$ and computing the corresponding function fitness values. The basic genetic unit of biological individual $x_i(t)$ is gene, and the genes are ranked in sequence to form chromosomes. The initial population[3] consists of a certain amount of chromosomes at the beginning of the search. In fact fitness function is the objective function of optimization problem, the evolutionary algorithm will search for the chromosome with maximum fitness.

Step 2: Selecting, crossing and mutating the individuals. The individuals with the better fitness replace the worse individuals, namely, the selection operation embodies the biological genetic process of fitted survival. Crossover operation is the exchanging of gene pieces of the male parent chromosome and female parent chromosome. Some parts of the genes are changed in mutation operation

Step 3: Computing the fitness values of the new individuals: the sub-generation generated by the male parent and the female parent forms a new generation (population), and then we compute the all fitness values of a new generation to prepare a new evolutionary operation.

Step 4: Stopping the operation when the most optimal fitness solutions are solved or stop criterions are satisfied, and then output the solutions of the chromosome; otherwise, return to step 1.

4 The Theory of Cyclic-Code Generated Polynomial

4.1 Field[1,4]

If F is a set included at least one element, and satisfy the follow criterions:

a. All elements of F constitute a plus Abel-group, and the plus unit element is marked 0.
b. All non-0 elements of F constitute a multiple Abel-group, and the multiple unit element is marked 1.
c. The elements of F satisfy the distributive and associative regulations between plus and multiple as follows:
   $a(b+c)=ab+ac$
   $(b+c)a=ba+ca$

where $a,b,c \in F$, so we call F a field.

4.2 $GF(q)$-Field

$GF(q)$ is a field which satisfies the following criterions

a. The $GF(q)$ includes q elements
b. All results calculated by modulo-q plus and modulo-q multiple with the elements of $GF(q)$ belong to $GF(q)$.

Example 1: Assume $GF(9)=\{0,1,2,3,4,5,6,7,8\}$, $\{8\}$ and $\{7\}$ belong to $GF(9)$, then (8+7) modulo 9=6 belongs to $GF(9)$, 8*7 modulo 9=2 belongs to $GF(9)$ too.

Example 2: Assume $p(x) = x^3 + x + 1$ is a polynomial that cannot be divided by other polynomial of $GF(2)$, then the set consisting of 8 leaves of polynomials

$$\{\overline{0}, \overline{1}, \overline{x}, \overline{x^2}, \overline{x+1}, \overline{x^2+x}, \overline{x^2+1}, \overline{x^2+x+1}\}$$

produced by modulo $p(x)$ form a finite set $GF(2^3)$ of 8 degrees.

Therefore, we know that the $GF(q)$ is a plus cyclic group and a multiple cyclic group.

4.3 Cyclic codes[1,6]

In N-dimensional linear space V_n in field $GF(q)$, if $V_{n,k}$ is a k-dimension subspace, and to any $C_i=(C_{n-1},C_{n-2},...,C_0)\in V_{n,k}$ there is $\overline{C_i}=(C_{n-2},...,C_0,C_{n-1})\in V_{n,k}$, so we call $V_{n,k}$ a cyclic subspace or cyclic code, where $GF(q)$ is finite field of q exponentials, it's clear that $V_{n,k}$ is a k-dimension linear subspace of n-dimension linear space.

From above, we know that cyclic codes in field $GF(q)$ are linear grouping codes which every code has cyclic feature. Therefore, we can get another definition of cyclic codes:

Assume that C is a linear and $a=(a_0,a_1,...,a_{n-1})$ is a legal code of C, a new code is formed by shifting the elements along one place and taking one off the end and putting it on to the beginning, then we get $b=aT=(a_1,a_2,...,a_{n-1},a_0)$, and b is a legal code of C too, so we call C a cyclic code.

For example, suppose a cyclic-code generated matrix and a checkout matrix of Hamming code C are as follow:

$$G=\begin{bmatrix}1000101\\0100111\\0010110\\0001011\end{bmatrix}\qquad H=\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix}=\begin{bmatrix}1110100\\0111010\\1101001\end{bmatrix}$$

From the first line of generated matrix based on left moving a place, we get 16 codes (1000101) (0001011) (0010110) (0101100) (1011000) (0110001) (1100010)
(0100111) (1001110) (0011101) (0111010) (1110100) (1101001) (1010011) (1111111) (0000000)

From these codes we can easily see that if C_i is the code of C, moving cyclic code leftward or rightward a place, we can also get the code of C. So this group of linear grouping code is cyclic code.

Assume $C=(C_{n-1},C_{n-2},...,C_0)$ is a code of $[n,k]$ cyclic codes, the corresponding polynomial is $C(x)=C_{n-1}x^{n-1}+C_{n-2}x^{n-2}+\cdots+C_0$. It is called the polynomial of code C (or code polynomial).

4.4 The cyclic-code generated polynomial [3,4,6,7]

Assume $f(x)$ an n -exponential polynomial on $GF(q)$, and $g(x)$ the first 1-coefficient factor of $f(x)$, namely, $f(x)=g(x)h(x)$.

A set: $I=\{m(x)g(x)\mid \partial m(x)\le n-1-\partial g(x)\}$, where ∂ is exponential numbers of polynomial. Then, we call $g(x)$ the generated polynomial of I, namely, I is generated by $g(x)$.

Assume $g(x)$ a cyclic code (polynomial), then we call it the cyclic-code generated polynomial. There are two important theories of solving legal codes as follows:

Theory 1: $C(x)$ is the cyclic legal code if and only if $C(\alpha_i)=0, i=1,2,\cdots,n-k$. Where α_i is the root of $g(x)$ $(i=1,2,\cdots,n-k)$

Theory 2: Assume $C(x)$ is the cyclic code, $c_{n-1},c_{n-2},\cdots,c_0$ are the corresponding coefficients of $C(x)$, and $C(\alpha_i)=0, i=1,2,\cdots,n-k$. Where α_i is the root of $g(x)$ $(i=1,2,\cdots,n-k)$, and $\alpha_i\neq\alpha_j$ $(i\neq j)$. Then, we can solve the values of $c_{n-1},c_{n-2},\cdots,c_0$ through solving

$$\begin{bmatrix}\alpha_1^{n-1},\alpha_1^{n-2},\cdots,\alpha_1,1\\ \alpha_2^{n-1},\alpha_2^{n-2},\cdots,\alpha_2,1\\ \vdots\qquad\qquad\vdots\ \vdots\\ \alpha_{n-k}^{n-1},\alpha_{n-k}^{n-2},\cdots,\alpha_{n-k},1\end{bmatrix}\begin{bmatrix}c_{n-1}\\c_{n-2}\\\vdots\\c_0\end{bmatrix}=0,$$

and the cyclic legal code is $(c_0,c_1,\cdots,c_{n-1})$.

Therefore, we can use the roots of cyclic-code generated polynomial $g(x)$ to construct the cyclic legal code C. But the difficulty in constructing cyclic legal code C is how to solve the roots of cyclic-code generated polynomial. But general method to solve the roots is to factor cyclic-code generated polynomial, as we know, this method is infeasible for solving the roots of high-exponential polynomial. To resolve this difficulty, in the next section we will propose a feasible method on how to use evolutionary computation to solve the roots of cyclic-code generated high-exponential polynomial

5 The Algorithm of Solving the roots of Cyclic-Code Generated polynomial by Evolutionary Computation

Assume $g(x)$ is a cyclic-code generated polynomial, then we convert the solving problem to a single-objective optimal problem as following,

$$\min g(x)$$

subject to $x \in GF(q)$ and $g(x) \geq 0$

Because the field $GF(q)$ is the modulo-q field, the method of finding optimal solutions is more difficult than classical method of evolutionary computation. Therefore, we have to improve the algorithm of evolutionary computation as following:

Step 1, initialize population P_0, set $t = 0$, produce n integral individuals $P_0 = \{\alpha^0_i\}$, $i = 1,2,\cdots N$ at random, and calculate the function fitness values of the corresponding individual, and sort the modulo-q values in the order from small to large.

Step 2, Select some individuals to form a reproduction pool

$P_1 = \{\alpha^1_i\}$, $i = 1,2,\cdots m$

Step 3, crossover the individuals of P_1

$$\alpha^2 = \text{int}(\sum_{j=1}^{m} \alpha_j^1 \beta_j)$$

where $\sum_{l=1}^{m} \beta_l = q$, $0 < \beta_l < q$ are random number, $l = 1,2,\cdots m$

Step 4, use new individual to mutate the parents. If the function fitness value of independent α^2 is better than the worst modulo-q value calculated by step 1, then, use α^2 to replace the corresponding individual, otherwise, return to step 3.

Step 5, If the fitness value of function $\|G(x)\| < 10^{-10}$ or $t > T$ is reached, output the modulo-q individuals, and stop the running. Otherwise, return to step 3.

6 Numerical Experiments

In this section, we apply evolutionary computation to solving the roots of cyclic-code generated polynomial in a special modulo-q field *GF(q)* to indicate that we can use this method to replace traditional method in which factor the polynomial by artificial operation to solve the roots in this complex modulo-q field *GF(q)*.

Experiment 1: Suppose the length of a code 14, namely, n=14, and the places of checking-code is 4,so *n-k* equals 10, the cyclic-code generated polynomial is as following:

$$g(x) = x^{14} + 4x^{12} + 7x^{10} + 6x^9 + 4x^3 + x + 1 = 0$$

subject to $g(x) \geq 0$ and $1 \leq x \leq 1000$, $x \in GF(2^3)$

we set maximum iteration times T=20000, after running 543 times the 10 roots are attained as in the following table

No.	Root	No.	Root
1	49	6	681
2	985	7	41
3	129	8	993
4	689	9	681
5	33	10	105

Experiment 2: Suppose the length of a code 14, namely, n=14, and the places of checking-code is 2,so *n-k* equals 12, the cyclic-code generated polynomial is as following:

$$g(x) = x^8 + 3x^7 + x^6 + 5x^3 + 3x^2 + 2x + 1 = 0$$

subject to $g(x) \geq 0$ and $1 \leq x \leq 1000$, $x \in GF(2^3)$

we set the maximum iteration times T=15000, after running 148 times the 12 roots are attained as in the following table

No.	Root	No.	Root
1	993	7	25
2	757	8	997
3	989	9	785
4	65	10	21
5	13	11	13
6	73	12	77

Through the analysis of the above experiments, we conclude that this method has at least two advantages, compared with the method of traditional manual factorizing polynomial. The first is that using evolutionary algorithm to solve the roots of cyclic-code generated polynomial can save much time, it only takes some minutes to solve the roots, which is impossible by factorizing the polynomial in such short time; the second is that this method can solve all the roots of the polynomial, while using the

method of factorizing the polynomial is difficult to solve all the roots of the cyclic-code generated polynomial, sometimes not even a single one.

7 Conclusions

This paper proposed an innovative method to solve the roots of the cyclic-code generated polynomial in the complex modulo-q field $GF(q)$ by using evolutionary computation. Through the experiments and the theoretical analysis, this method is the best method to solve the roots of this type of polynomials up to now, and authors haven't found anyone who proposed such high performance method. Therefore this method will change the method that only factor the cyclic-code generated polynomial of high exponential to find the roots manually in modulo-q field $GF(q)$, and it will also resolve the difficult problem of having roots but could not be solved. So, it extended the application of evolutionary computation in computer security, it also enhanced the reliability of the trustworthy computing at the same time.

References

[1] Wang Xingmei, Zhang Huanguo, Ma Jianfeng, Tan Zhongping, 1999. "Error-Correction Code Technology in Computer". People's Post Press.

[2]. Pan Zhengjun. Kang Lishan, Chen Yuping, Evolutionary Computation, Tsinghua University Press, 1998.

[3] Ye Dacheng, 1996, 24(12). "The solution to genetic algorithm of route selection and volume allocation problem in computer communication network". Electronic Journal.

[4] Wang Xinmei, 1991, 280 ~ 284. "Error-Correcting Code—Theory and Method". Xian: Xidian University Press.

[5] Holland J H. 1975, "Adaption in Natural and Artificial Systems". Ann Arbor: University of Michigan Press.

[6] Spillman R. 1993, 17(4), "Cryptanalysis of Knapsack Chipers Using Genetic Algorithms". Cryptologyia,

[7] Zhang Muxiang, 1994, 76(3), "Simulated Annealing Approach to the Minimum Distance of Error-Correcting Codes". Int J Electronics.