The Solution for Satisfaction of Maximal Boolean Polynomial Equations Based on Genetic Algorithm

Shunyu Yang

Zhengzhou Institute of Information Science and Technology, Zhengzhou 450001, China.

chaserysy@163.com

Keywords: Boolean polynomial equation, Optimization, Genetic Algorithm, Fitness function.

Abstract. In this paper, the solution for the satisfaction of Maximal Boolean Polynomial Equations based on Genetic Algorithm is proposed. The Genetic Algorithm is used to model the problem in this method, and the population of the Genetic Algorithm is taken as the solution space of the equation set and the individual is a feasible solution of the equation set. By applying the binary fitness code to the individual and selecting the appropriate individual fitness evaluation function, the more the number of the individual which make more equations with the value of 0 in the Boolean system is, the higher the corresponding fitness value is. After the selection operation, crossover operation and mutation operation, individuals with higher fitness are retained and inherited to the next generation until the algorithm converges and eventually obtains the approximate optimal solution of the problem. Using matlab to program, the optimal solution obtained in the experimental examples is 182.

1. Introduction

The Boolean system is a system of equations consisting of Boolean equations [1]. In the case of the satisfiability problem (Max-PoSSo problem) of the maximal Boolean polynomial equations, given m Boolean polynomials with n variables:

$$\begin{aligned} &f_1(x_1, x_2, ..., x_n) \\ &f_2(x_1, x_2, ..., x_n) \\ &\quad ..... \\ &f_m(x_1, x_2, ..., x_n) \end{aligned} \tag{1}$$

Try to find a set of $X_1, X_2 \ldots\ldots X_m$ in GF(2) (each X_i value is in GF(2)) to make the Boolean equations $f_1, f_2 \ldots f_m$ have the largest number of values 0 in this group of variables [1].

In algebraic attacks of cryptanalysis, the attacker first sets the secret information (such as the key) as a variable, then establishes the corresponding polynomial equations through the relationship between the known information and the secret information, and resolves the secret by solving the polynomial equations. The solving of the Boolean system is of great significance in many aspects, such as the prevention of algebraic attacks, digital logic design, artificial intelligence [2].

2. Algorithm

2.1 Genetic Algorithm

Genetic Algorithm is a computational model that simulates the natural selection and the biological evolution of the genetic mechanism from Darwin's theory of biological evolution. It is a method of searching the optimal solution by simulating the natural evolutionary process. The initial population selects the individual according to the fitness function, and obtains the optimal individual in the last generation after crossover operation and mutation operation. Then decode it as the approximate solution of the problem. The elements of Genetic Algorithm are defined as follows:

Population: Consists of a certain number of individuals encoded by the gene, containing a set of multiple solutions of the Boolean system. Population size is the number of the solution in the solution space.

Individual: The entity with the characteristic chromosome.

Fitness Function: Also called the evaluation function. It is an indicator to judge the degree of the individual in the population. The function evaluates according to the objective function of the problem.

Selection Operation: Select a certain number of individuals each time from the population according to a certain probability, to prepare following crossover and mutation operations.

Crossover Operation: Some genes of the selected parental chromosome are mated and recombined to form the new individuals.

Mutation Operation: Select a mutation position randomly from the chromosome, performing 0 and 1 exchanges in binary encoding.

2.2 Variables Description

In the Genetic Algorithm modeling process, the variables used are shown in Table 1.

Table 1 Variables description

Variables	Description
M	Population
Xi，i=1,2,…,N	Individuals in the population
Ai，i=0,1,2,…,127	The i-bit of the individual code
N	Population size
Pr	Selection probability
Pc	Crossover probability
Pm	Mutation probability
F(Xi)，i=1,2,…,N	The fitness value of individual Xi
Fitness(Xi)，i=1,2,…,N	The relative fitness value of individual Xi
len_i	The number of the equations with the value of 0 corresponded to the individual Xi
maxlen	The len value of the optimal individual in the current population
minlen	The len value of the worst individual in the current population

2.3 Model Establishment

Based on the first session of the First National University of Mathematical Challenge, according to the above description of the Genetic Algorithm, we can find that the Max-PoSSo problem can be solved by the modeling of the Genetic Algorithm. Through the further analysis of the problem, combined with the concept of Genetic Algorithm and the elements, the problem is modeled as follows:

Population M: Using a method of randomly initializing the population, the initial population size is set to 100, contains 100 individuals.

Individual X: Individuals use binary coding, and the digital string which is composed of the independent variables $A_0A_1 \ldots A_{127}$ forms the individual. The coding sequence of a single individual represents a solution of the system, multiple individual forms the population. After several generations of evolution to get the best individual, it's coded value is used as the approximate optimal solution of the system.

The correspondence between the populations M, individual X and the solution space of the system, single solution are as follows:

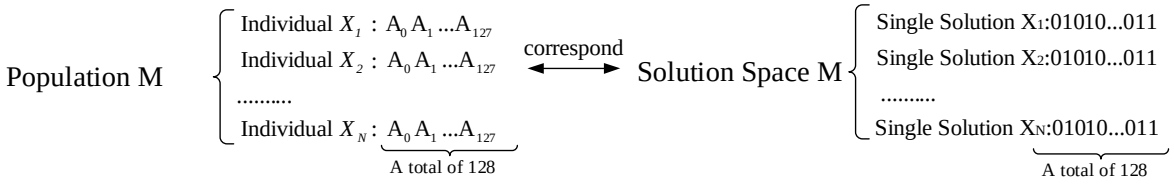

Fig. 1 The Correspondence between population and solution space, individual and single solution

Fitness Function F(X): For a single individual $Xi=A_0A_1 \dots A_{127}$, implant the corresponding independent variables into the Boolean equations, the more the number of the equations with the value of 0, the greater the fitness value of the individual is, so the greater the probability of being selected in the genetic operation. Individuals that have been retained by multiple iterations are those with high fitness values. The fitness function is calculated as follows:

The fitness value F(X_i) of individual X_i=A0A1...A127 ⟷ correspond ⟷ Substituting X_i=A0A1...A127 into the system, calculate len_i and F(X_i)

Fig. 2 Individual fitness value correspondence

The individual X_i fitness function is defined as follows (after normalization):

$$\mathrm{F}(X_i) = 1 - (1 - \frac{len_i - minlen}{maxlen - minlen + \varepsilon})^m \tag{2}$$

$\mathrm{len_i}$ represents the number of the equations with the value of 0 corresponded to the individual X_i in the current population; minlen represents the least number of the equations with the value of 0 in all individuals of the population; maxlen represents the maximum number of the equations with the value of 0 in all individuals of the population; ε represents a very small number, used to ensure that the denominator is not 0, is taken as 0.0001 in this problem; m is the normalized accelerated phase-out index, which accelerate the speed of genetic evolution.

Selection Operation: There are multiple selection strategies in practice, and this problem uses the most commonly used Roulette wheel selection method. In this method, the probability that an individual is selected depends on the relative fitness of the individual, individual X_i relative fitness is defined as:

$$\mathrm{Fitness}(X_i) = \frac{F(X_i)}{\sum_{j=1}^{\mathrm{N}} F(X_i)} \tag{3}$$

$\mathrm{Fitness}(X_i)$ indicates the relative fitness value of individual X_i, $\mathrm{F}_{(X_i)}$ indicates the individual fitness, N is the population size.

The basic idea of the Roulette wheel selection method is to divide a turntable into n sectors based on the selection probability $\mathrm{P}_{(X_i)}$ of each individual, the center angle of the i-th sector is:

$$2\pi \frac{f_{(x_i)}}{\sum_{j=1}^{n} f_{(x_j)}} = 2\pi P_{(x_i)} \tag{4}$$

Crossover Operation: Using the Binary single point intersection, the single point intersection is randomly set an intersection in the two parent individuals $\mathrm{X_i}$ and $\mathrm{X_j}$ coding strings, then exchange the front or rear part of the intersection, generate two new offspring individuals, and add to the population. Assume that the selected individuals X_i and X_j in the population are encoded as follows:

$$X_i = a_1 a_2 \dots a_k a_{k+1} \dots a_n \tag{5}$$

$$X_j = b_1 b_2 \dots b_k b_{k+1} \dots b_n \tag{6}$$

Randomly select the K-th position as the intersection, crossover the gene after the K-bit , and the results are shown in figure 3.

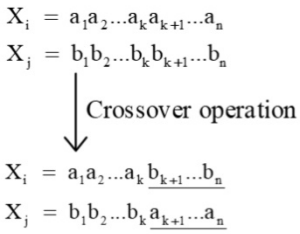

Fig. 3 Crossover operation

Mutation Operation: Randomly select a mutation position on the individual coding sequence, exchange 0 and 1 with each other, and the new individual is produced. The mutation operation can maintain the population diversity and enhance the local random search ability of the algorithm.

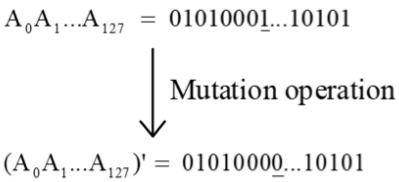

Fig. 4 Mutation operation

After using the Genetic Algorithm to model the problem, the Max-PoSSo problem can be transformed into the use of Genetic Algorithms to find the optimal solution in the solution space.

3.Experiment

3.1 Experiment Process

Based on the first session of the First National University of Mathematical Challenge, according to the established Genetic Algorithm model above, this paper designs the following algorithm:

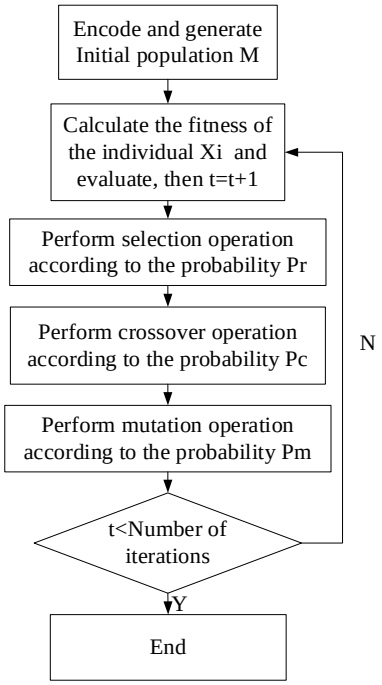

Fig. 5 Algorithm process

Step1: Select the encoding strategy. The binary coding strategy is used to encode the single individual according to the form of 0, 1 string. The individual code in the initial population is

randomly generated.

Step2: Defining the genetic strategy. Population size is M, determine the selection probability P_r, crossover probability P_c, and Mutation probability P_m.

Step3: the number of iteration t is set to zero, and M initial individuals are randomly generated according to population size. Then initialize the population.

Step4: Define the calculation method of the fitness function F(X).

Step5: Calculate the fitness value F (Xi) of each individual in the population.

Step6: Use the selection operator to select individuals from the population.

Step7: Crossover the selected individual according to the crossover probability P_c.

Step8: Make the individual mutation operation according to the mutation probability P_m..

Step9: Determine whether t has reached the number of the iteration, if it has not been reached, t=t+1, and return step5, or preserve the optimal individuals in the current population as the approximate optimal solution of the problem and end the program.

According to the above algorithm to solve the model, the specific result is in the next section.

3.2 Results and Analysis

The Genetic Algorithm is used to establish and solve the model, the initial population size is set to 100, the individual code length is set to 128, the crossover probability is set to 0.6, the probability of mutation is set to 0.1. The number of iteration is set to 200. Due to the randomness of the initial population, the approximate optimal solution of each program running is not the same. One of the results of the operation shown in figure 6.

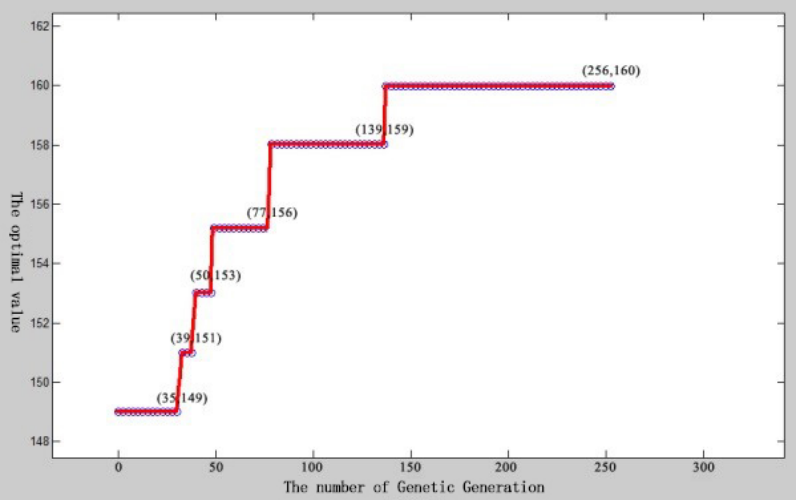

Fig. 6 Genetic Algorithm operation results

After the analysis of the figure is easy to draw, the optimal value of the Genetic Algorithm operation results, with the increasing of the genetic algebra(X axis), is constantly increasing in the overall trend (Y axis). In other words, under the action of genetic evolutionary strategy, the individual's fitness value of every generation is getting higher and higher, the optimal value is rising. The optimal result in figure 6 finally converges to the point (256, 160), indicating that the Genetic Algorithm ends at the 256th generation, the maxlen corresponded to the optimal individual is 160, which means that the optimal individual can make 160 equations of the equation set equal to zero.

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
78	len_1(i,1)=myLength(popm(i,:))...	603	9.318 s	39.3%	
135	len(i,1)=myLength(popm_sel(i,:...	606	9.301 s	39.2%	
50	len(i,1)=myLength(popm(i,:));%...	200	4.728 s	20.0%	
194	axis([0 C_old 0 200]);	3	0.112 s	0.5%	
189	hold on	3	0.044 s	0.2%	
All other lines			0.196 s	0.8%	
Totals			23.698 s	100%	

Fig. 7 Main program time consumption

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
146	char(112)= ((x111+x101+x99+x78...	51609	10.045 s	1.2%	
194	char(160)= ((x113+x112+x111+x1...	51609	9.886 s	1.2%	
247	char(213)= ((x115+x114+x113+x1...	51609	9.869 s	1.2%	
170	char(136)= ((x112+x111+x110+x1...	51609	9.808 s	1.2%	
74	char(40)= ((x108+x107+x106+x97...	51609	9.707 s	1.2%	
All other lines			759.533 s	93.9%	
Totals			808.847 s	100%	

Fig. 8 Single Boolean polynomial time consumption

The time consumption of the Genetic Algorithm operation is shown in figure 7 and figure 8. We can draw from the figure above that the main run time of the algorithm is spent on the calculation of Boolean polynomial equations, and the 146,194,247 equations have the longest computation time in all equations and can be defined as the complex equations.

Because of the "premature phenomenon" of the Genetic Algorithm, the local convergence rate is slow in the absence of effective heuristic information, and the optimal result is usually the local optimal result, which is not the globally optimal solution. In order to overcome this problem, we have done several fine tunings of parameters in the experiment, repeated experiments, and achieved some intermediate results. Some of more optimized results are in Table 2.

Table 2 Optimization results

The maximum value	Coded value
maxlen=154	1010011111111010110011001100010101111111010011000100000111100110 0101111111000101100110101111010010110111101111010011111010010101
maxlen=167	1000100101000100100111100100010000100010010010110110101100110010 0111100110110111111011000111000101001101101010001100111100011010
maxlen=170	0001000000010101101010110100000011010000111100100111010100100011 0001001000101110111101000110000110011110110111011110010111001000
maxlen=175	1101100100101111000111000000000000000000100101100100101011100000 0000000000000000000000000000000000000001100001100010010100011111
maxlen=182	1010111010001001110010000110000000001000000000100000000000000000000 0000000000000000000000000000000011111111111111111111111111111111

In the results of the experiments above, we obtained the optimal value of 182, the corresponding individual code is：

0001000000010101101010110100000011010000111100100111010100100011
0001001000101110111101000110000110011110110111011110010111001000

We verify the result by substituting the value of the independent variable corresponded to the individual into the system, the number of equations with the value of 0 is 182, that is, which means 182 is the correct result to satisfy the condition.

4. Conclusion

In this paper, the Max-PoSSo problem is effectively modeled by Genetic Algorithm. The population of the Genetic Algorithm is taken as the solution space of the system, and the binary coded individual is a feasible solution of that. By selecting the appropriate fitness function, the higher the number of equations with the value of 0 in the system, the higher the fitness value of the corresponding individual is. After the selection operation, crossover operation and mutation operation, the individual with higher fitness is retained and the optimal solution of the problem is obtained.

References

[1] Li Xiaowei. Analysis of Nonlinearity of Boolean Functions and Applcation in Cryptography [D]. Chengdu University of Technology, 2012.

[2] Chen Huajin. Construction and Analysis of Boolean Functions Resistant to Algebraic Attacks [D]. PLA Information Engineering University, 2013.

[3] C.E. Shannon. Communication theory of secrecy systems [J]. Bell Technical Journal, 1949, 28(4): 656-715.

[4] Zhang Jianming, Shen Shengyu, Li Sikun. Algorithms for Deriving Minimum Unsatisfiable Boolean Subformulae [J]. Acta Electronica Sinica, 2009, 37(5):993-999.

[5] Shuai Xunbo, Ma Shunan. Solving N-queen Algorithm based on boolean genetic operator [J]. Computer Engineering and Applications, 2011, 47(16): 49-51.

[6] Li Yunqiang, Sun Huaibo, Wang Ailan. Fast algorithms for polynomial representations of boolean functions and pseudo-boolean functions [J]. Computer Engineering and Applications, 2007, 43(1): 50-52.

[7] Wang Peigen. Boole Function and Boole Polynomial [J]. Journal of Capital Normal University (Natural Science Edition), 2006, 27(5): 15-18.

[8] Li Xin, Lin Dongdai, Xulin. A High Efficient Boolean Polynomial Representation [J]. Journal of Computer Research and Development, 2012, 49(12): 2568-2574.

[9] Huang, Z. and Lin, D.: A New Method for Solving Polynomial Systems with Noise over F2 and Its Applications in Cold Boot Key Recovery, Selected Areas in Cryptography, LNCS 7707, pp 16-33, 2013.

[10] Bard G V, Courtois N T, Jefferson C. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF (2) via SAT-solvers. 2007.