Research on Automatic Test Paper Composition System Based on Improved Genetic Algorithm

Xuecheng Tong, Botao Liu*, Wentao Li, Hao Xiao, Tao Xu
Yangtze University, Hubei, Jingzhou 434000, China.

Abstract

The test paper system module, to achieve automatic test paper, and requires the combination of the test paper to achieve all the indicators, such as the basic constraints of the test paper, difficulty control, question control have been met. According to the characteristics of this problem, a mathematical model is established to obtain an appropriate fitness function. This paper presents an effective method of genetic algorithm to solve the problem of paper grouping. The experimental results show that the experimental results obtained by using genetic algorithm meet the requirements and have good practicability.

Keywords

Test Question Bank; Automatic Volume Grouping; Genetic Algorithm; Fitness Function.

1. Introduction

Automatic test paper composition is based on the requirements of teachers and teaching, the computer automatically selects test questions from the test question bank to form a test paper that meets the requirements. It is an important part of Computer Assisted Instruction (CAI). Commonly used automatic test paper generation methods can be roughly divided into two categories:

1) Random extraction [1,2], That is, according to the control index of the state space of the test paper, the computer randomly selects a test question that meets the control index and puts it into the test question bank of the test paper. This process is repeated until the test paper is completed or the test questions that meet the control index cannot be extracted from the test bank. This method is simple in structure, has great arbitrariness and uncertainty, cannot grasp the changing requirements of the question bank as a whole, and is not intelligent.

2) Backtracking [3,5], That is to record each state type generated by the random extraction method, and release the state type recorded last time when the search fails. Then change a new state type according to a certain rule to test, and through continuous backtracking until the test paper is generated or returned to the starting point, it is to use the backtracking method to realize automatic test paper. Practice has proved that this method is suitable for a question bank system with a relatively small type and volume of questions.

In actual application, the program structure is relatively complicated, and the randomness of the selection of test questions is poor, and the time for generating papers is long. For the more and more popular test process in which test takers randomly adjust questions immediately, it no longer meets the requirements. GA (Genetic Algorithms) [6,7] is a computational model that simulates the evolution of natural organisms. The genetic algorithm can solve the blind randomness of the first method mentioned above, and can select individuals that meet the conditions from the group. It has strong intelligence and can produce different offspring according to different environments. It is dynamic and adaptive. It can meet the changing requirements of the question bank. Therefore, when the test question library has a large capacity, wide coverage, and a large amount of calculation for
selecting questions, the inherent parallelism of genetic algorithms can be used to effectively solve the problem of automatic test paper generation.

2. Problems of generating test paper

The automatic test paper composition is based on the constraints given by the examiner (such as the total number of test questions, the number of individual types of questions, the score of each question, the total score of the test paper, the distribution of chapter knowledge points, the degree of difficulty, the proportion of question types, etc.). Search the test questions matching the characteristic parameters in the test question library to extract the best combination of test questions.

2.1 Problems of generating test paper

Each test paper is designed as a chromosome, and each question is designed as a gene. Since the title of the test paper is fixed and the score is also fixed, at the beginning, we initialize the population and only need to meet the knowledge points obtained the corresponding type of questions and the total score of all the questions, and these two conditions can get a complete chromosome. Design the chromosome code according to the model.

Binary coding is used in traditional genetic algorithms. When using binary coding, every question in the question bank must appear in this binary bit string, 1 means the question is selected, 0 means the question is not selected. Such a binary bit string is relatively long, and when performing crossover and mutation genetic operators, the number of questions of various question types is inconvenient to control.

Therefore, the real number coding scheme is adopted here. A test paper is mapped to a chromosome, and the question number of each question constituting the test paper is used as a gene. The value of the gene is directly represented by the question number. The question numbers of each question type are put together, divided by question type, and then the operation of the genetic operator is carried out by paragraph, which ensures that the total number of questions for each question type remains unchanged. There are a total of n test questions and k types of test papers. The n test questions are divided into k sections according to the question type, and each section represents 1 question type. If the number of questions contained in the r-th (r=1, 2, …, k) question type is ri, the chromosome is:

$$sn_1(1)sn_2(2)…sn_1(r_1)sn_2(2)…sn_2(r_2)…sn_k(1)sn_k(2)…sn_k(r_k)$$

The initial population of the test paper is not generated by a completely random method, but randomly generated according to the requirements of the total number of questions, the number of individual question types, and the total score of the test paper, so that the initial population meets the requirements of the number of questions and the total score from the beginning. This speeds up the convergence of the genetic algorithm and reduces the number of iterations. The use of grouped real number coding can overcome the shortcomings of the excessive search space and long code length of the previous binary coding, and at the same time cancel the individual decoding time and improve the solution speed [8].

2.2 Fitness function design

The fitness function is an index used to judge the pros and cons of individuals in the test paper group [9]. The genetic algorithm uses the fitness value to guide the search direction, without the need for continuous or derivable fitness functions and other auxiliary information. Because the number of questions and total score requirements have been considered when initializing the population, only the distribution of knowledge points and the difficulty coefficient are left to consider. So the fitness function is related to the difficulty coefficient of the test paper and the distribution of knowledge points. The formula for the coefficient of difficulty of the test paper is shown in formula 1 (1).

$$p = (\sum D_i \times S_i) / \sum S_i \quad (i = 1, 2, \ldots, N)$$

Among them, N is the number of questions contained in the test paper, Di and Si are the degree of difficulty and score of the i-th question respectively. The distribution of knowledge points is measured
by the coverage rate of an individual knowledge point. For example, it is expected that the test paper contains N knowledge points, and the union of all topic knowledge points in an individual contains M (M<=N), then the coverage of knowledge points The rate is M/N. The smaller the difference between the user's expected difficulty coefficient EP and the test paper difficulty coefficient P, the better, and the greater the coverage of knowledge points, the better, so the fitness function is shown in formula (2).

\[
f = 1 - \left(1 - \frac{M}{N}\right) \times f_1 - \left|E_P - P\right| \times f_2
\]

(3)

Among them, f1 is the weight of the distribution of knowledge points, and f2 is the weight of the difficulty coefficient. When f1=0, it degenerates to only limit the difficulty coefficient of the test, and when f2=0, it degenerates to only limit the distribution of knowledge points.

3. Algorithm solution

In summary, the overall flow chart of the application of genetic algorithm in this system is shown in Figure 1.
3.1 Initial population

Since the chromosome model is explained above, the order of the questions in the test paper is certain, that is, multiple-choice questions, fill-in-the-blank questions, true or false questions, and finally the kind of questions that require calculation or explanation. So each test paper is this template, and the population size is given based on experience or experimental test data.

The initialization strategy is as follows: The test paper strategy is the basis of the automatic test paper algorithm, and refers to the organization and manifestation of the test paper control parameters. The test paper control parameters selected by this system are: test paper difficulty (P), test paper discrimination (D), total score value of test paper (W), estimated time of test paper (T), test question type $j$ ($j = 1, 2, 3, 4, 5$, numbers 1–5 represent 5 different question types), the number of questions in the test paper (N), the number of questions in each question type (TNJ), and the score of each question type (TWJ)[10].

$$P = \sum_{j=1}^{5} (TP_j \times TW_j) \div W, \quad D = \sum_{i=1}^{N} (Di \times Wi) \div W$$  \hspace{1cm} (4)

Among them, TPJ represents the difficulty of each question type. Based on past experience in marking papers, the difficulty of each question type also has a greater impact on the quality of the test paper. For example, multiple-choice questions and fill-in-the-blank questions are not easy to be too difficult, otherwise the average test paper is often lower. Therefore, not only the restriction on TPJ is added to the volume target, but the TPJ parameter is also introduced in the calculation of P. Its calculation formula is as follows:

$$TP_j = \sum_{i=1}^{N} (P_i \times W_i) \div TW_j, \quad i \not\in j$$  \hspace{1cm} (5)

The volume target is as follows: The actual number of test questions in the test paper is equal to the number of test questions (NR) required by the test paper, namely.

$$NR = N = \sum_{j=1}^{5} TN_j$$  \hspace{1cm} (6)

The actual total score of all question types in the test paper is equal to the total score (WR) required by the test paper, namely

$$WR = W = \sum_{j=1}^{5} TW_j$$  \hspace{1cm} (7)

3.2 Selection operator

The function of the selection operator is to decide whether the next generation is eliminated or not according to the individual's pros and cons. Through selection, individuals with high fitness have a greater chance of survival, but individuals with low fitness may not necessarily be eliminated because of fitness Low individuals may also have high-quality genes. This system uses the roulette method, which is currently the most commonly used and classic selection method in genetic algorithms. In this way, some individuals with low fitness can be retained. The realization process is as follows: first calculate the function fitness of each parent individual, then sort from small to large and calculate its roulette probability. Finally, by randomly generating real numbers between 0 and 1, the newest parent individual corresponding to the roulette probability is selected [11]. The specific implementation formula is: assuming that the population size is n and the fitness of individual i is $f_i$, the probability of individual i being selected to be inherited to the next population is:

$$P_i = f_i \div \sum_{i=1}^{n} f_i$$  \hspace{1cm} (8)
3.3 Crossover operator
We have determined what the chromosomes look like during the modeling. What we can ensure is that in the same position on different chromosomes, the question types are the same, but the scores and difficulty coefficients of the questions may be different. Crossover can be divided into Single-point crossover and multi-point crossover, multi-point crossover as the name implies, there are multiple changes on the same chromosome. Because the crossover points are random, a single chromosome crosses multiple times, which is actually a multipoint crossover. So here we use single point crossover. In order to ensure a greater possibility of crossover success, two genes at the same position on different chromosomes are selected for crossover, so as to reduce the dilemma of crossover failure due to mismatched scores.

3.4 Mutation operator
In the genetic algorithm, the probability of mutation is relatively low. Set the mutation rate by setting the constant mutationRate. Within the range of [1,n] (within the entire test paper), randomly find a chromosome mutation point (single item), and find a mutation gene in the database to replace this mutation point according to certain rules. The selection rule for mutation is: because every gene is a topic, we must ensure that the type of the topic remains the same when we mutate. Of course, the score cannot be changed. If the score changes, then the entire gene does not conform to the rules. So, the principle of variation---the difficulty coefficient and knowledge points of the subject are changed, and the others are immutable.

4. Test
In the case of all conditions being unchanged, multiple calls, the results appear more satisfactory, and

```
// Expected fitness
double EXPAND_ADAPTER=0.9
TempExam tempExam=generatePaperService.getResult( count=28, expExam, q_subject="data structure",EXPANDE_ADATPER);
```

Figure 2. Expectation setting

According to the above formula f=1-(1-M/N)*f1-[EP-P]*f2 configuration f1,f2 These two scale factors can be changed at will. If the test paper focuses on knowledge points, f1 is larger than f2. If you pay more attention to the control of the difficulty coefficient, then f2 is slightly larger. Dynamic trade-offs

```
Total score list=+++++180+++++
Total score list=+++++180+++++
Fitness for obtaining test papers=99.95
The overall difficulty of obtaining the test paper=0.7
Single item assessment=Question type:Multiple choice degree of difficulty:8.62 Topic covers knowledge points:Linked list
Single item assessment=Question type:Multiple choice degree of difficulty:8.8 Topic covers knowledge points:Linked list
Single item assessment=Question type:Multiple choice degree of difficulty:8.67 Topic covers knowledge points:Figure
Single item assessment=Question type:Multiple choice degree of difficulty:8.5 Topic covers knowledge points:Figure

Figure 3. First run result
```

```
Total score list=+++++180+++++
Fitness for obtaining test papers=99.94
The overall difficulty of obtaining the test paper=0.63
Single item assessment=Question type:Multiple choice degree of difficulty:8.77 Topic covers knowledge points:Figure
Single item assessment=Question type:Multiple choice degree of difficulty:8.57 Topic covers knowledge points:tree
Single item assessment=Question type:Multiple choice degree of difficulty:9.01 Topic covers knowledge points:tree
Single item assessment=Question type:Multiple choice degree of difficulty:8.01 Topic covers knowledge points:Figure
Single item assessment=Question type:Multiple choice degree of difficulty:9.55 Topic covers knowledge points:Stack

Figure 4. Results of the second operation
```
Some optimizations have been made in the program to ensure that the given knowledge points will be used. The method used is to compare the knowledge point coverage of each test paper. If there is something missing from the set time, then it will be changed when it is changed. The direction variation of all missing knowledge points. Of course, the mutation rate can also be adjusted.

<table>
<thead>
<tr>
<th>f1(Knowledge point weight)</th>
<th>f2(Difficulty factor weight)</th>
<th>Overall difficulty of the test paper</th>
<th>Overall fitness of the test paper</th>
<th>Target fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>0.631</td>
<td>0.913</td>
<td>0.9</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>0.625</td>
<td>0.925</td>
<td>0.9</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.628</td>
<td>0.999</td>
<td>0.9</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>0.627</td>
<td>0.951</td>
<td>0.9</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3</td>
<td>0.604</td>
<td>0.956</td>
<td>0.9</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>0.643</td>
<td>0.978</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Figure 5. Summary of operation results under different coefficients

The difficulty coefficients in Figures 2, 3, 4, and 5 are all around 0.75. Some basic tests have been done above. The overall difficulty of the test paper also depends on the difficulty of the questions in the question bank. The overall difficulty in this test is about 0.6, which means that the overall questions in the question bank that meet our conditions are like this. If you want to customize the difficulty coefficient, it is also possible to set it through setDifficultyLevel in the expected test paper. If it is set, you can directly choose to fluctuate around ±0.15 within the set value range when the difficulty coefficient is directly selected when initializing the population. The final result is still impressed by the difficulty of the overall database test questions.

5. Summary

In this paper, the improved genetic algorithm is used to solve the problem of automatically generating test papers, breaking the traditional method of searching by binary code. According to the total number of questions, the number of single question types, and the total score of the test paper, these restrictions make the obtained chromosomes can be satisfied from the beginning. The basic requirement of the test paper is that you only need to control the degree of difficulty of the entire chromosome and the coverage of knowledge points in the later stage, thus overcoming the shortcomings of the previous search by binary code, the space is too large and the code is too long, and there is no need to decode again, causing additional This method improves the overall performance of the system. Experiments show that compared with the traditional genetic algorithm, this method can obtain the target test paper faster in the same question bank, the knowledge point coverage of the test paper and the overall difficulty of the test paper Both are better than expected. Compared with traditional genetic algorithm test papers, the quality of the test papers obtained is better than that.

Acknowledgments

This research is supported by three projects: The project of young people in the Education Hall of Hubei (No.Q20161311); The Yangtze Youth Fund (No.2015cqy53); The Yangtze University Students' Innovation and Entrepreneurship Training Program Project (No. 2019182).

References


