Journal of Physics:
Conference Series

PAPER • **OPEN ACCESS**

Intelligent Test Paper Generation Based on Dynamic Programming Algorithm

To cite this article: YiFei Wang and SuRong Wang 2020 *J. Phys.: Conf. Ser.* **1682** 012023

Intelligent Test Paper Generation Based on Dynamic Programming Algorithm

YiFei Wang[1],SuRong Wang[2]

[1]ResearchInstitute of Economics and Management,Southwestern University of Finance and Economics,Chengdu610074，China；

[2]School of International Economics and Trade, Shanghai University of International Business and Economics，Shanghai 201620，China

yifeiiris_wang@smail.swufe.edu.cn

Abstract. This paper describes the problem of intelligent paper grouping and its mathematical model. By optimizing and improving the traditional dynamic programming algorithm, its space complexity is reduced from O(nb) to O(b). At the same time, the flexibility of dynamic programming algorithm is increased by using marker function and tracking algorithm, and the result composition is tracked to obtain the optimal solution. Finally, through several experiments, the improved dynamic programming algorithm is compared with the greedy algorithm and brute force algorithm, and it is found that the improved dynamic programming algorithm has a very good result and is with high efficiency when applied to the simple test paper. It is the most recommended algorithm among the two algorithms compared in this paper.

1. Introduction

In any education system, examination has an irreplaceable role, it can objectively evaluate students' mastery of knowledge and skills. The analysis of test results can objectively reflect the quality of teaching [1].At present, many large-scale examinations at home and abroad, such as domestic academic examinations, grade certificate examinations [2], foreign TOEFL, GRE, GMAT, etc. [3], due to the popularization of computer technology, the technical means of these examinations have undergone tremendous changes. The development of traditional paper-based examinations into computer-assisted examinations and then into information-based online examinations has also led other industries to independently develop their own examination management software.When computers were used for teaching and research, the test paper system also appeared, but the test paper system at that time was only suitable for small test question banks.Since the 1970s, artificial intelligence has also participated in computer-assisted teaching, and the test paper system has been further developed.At present, many techniques such as random method [4], backtracking method [5], ant colony optimization algorithm [6] [7], etc. have been applied to the test paper system and have achieved certain results. In addition, genetic algorithm [8] and differential algorithm [9] are widely used test paper algorithms, but they are prone to premature convergence, and in the later stages of evolution, search efficiency is low.Some scholars have improved the genetic algorithm to improve the success rate and convergence speed of test papers [10]. Some scholars have improved the differential evolution algorithm to make it have better global optimization capabilities. However, there is still a lot of room for improvement in the exploration of intelligent test paper system and the application of test

paper algorithm.Aiming at the advantages and disadvantages of the above algorithms, this paper proposes a method of using dynamic programming algorithm to generate test papers. Although this type of algorithm is rarely used in the related research of test paper formation [11], the experiments in this paper show that when it is applied to simple test paper formation problems, the effect is very good and has good practicability.

2. Design of intelligent test paper system

2.1. Build model

Suppose our goal is to design a set of test papers that can exercise students abilities， and it is known that there are n questions to choose from, and the difficulty and average solving time of each question are diffi and ti respectively. Because the test time is limited, the maximum solution is set 'The question time is b minutes'. The question is how to choose the test questions, so that the difficulty of the test paper is maximized if the average problem solving time does not exceed b.

First, we model the problem. Assuming that the solution of the problem is <x1, x2,..., xn>, where xm is a 0-1 variable.When xm=0, it means that the question is not put in the test paper, when xm=1, it means that the question is put in the test paper.The objective function and restrictionsare as follows:

$$\text{Objective function: } \max \sum_{m=1}^{n} d_m x_m \quad (1)$$

$$\text{Restrictions: } \sum_{m=1}^{n} t_m x_m \le b,\ x_m \in N \quad (2)$$

Among them, dm is the difficulty of each test question, xm is a variable of 0-1, which represents whether the test question is put in the test paper, tm is the average problem-solving time of each problem, and b is the maximum problem-solving time set by the exam.

By taking the optimal solution between putting in and not putting in question i, the recurrence equation of the test paper problem is obtained.

$$\begin{aligned} &test_i(j) = \max\ \{test_{i-1}(j),\ test_{i-1}(j-t_k) + diff_i\} \\ &\text{s.t.} \\ &test_0(j) = 0,\ 0 \le j \le b; \\ &test_i(0) = 0,\ 0 \le i \le n; \\ &test_i(j) = test_{i-1}(y),\ j<t_i; \end{aligned} \quad (3)$$

2.2.Analysis and optimization of test paper problems

Using this method, there are two levels of FOR loops, so the time complexity is O(n×b) and the space complexity is O(n×b). However, in order to make the algorithm take up less time and space, it is very necessary to optimize the algorithm. Although time complexity is difficult to optimize further, there is still room for improvement in space complexity.

The recursive equation after optimizing the space complexity is as follows:

$$\begin{aligned} &test_i(j) = \max\ \{test_i(j),\ test_i(j-t_k) + diff_i\} \\ &\text{s.t.} \\ &test_0(j) = 0,\ 0 \le j \le b; \\ &test_i(0) = 0,\ 0 \le i \le n; \end{aligned} \quad (4)$$

At this time, the space complexity is optimized to O(b).

2.3.Flexible expansion

The above is only the calculation of the maximum achievable complexity value, but considering that

people may wish to master more information, this article adds a marking function and tracking algorithm to the original code.

The marking function means that the test paper includes the first i questions, the average available time does not exceed j, and the last question number included in the test paper when it reaches the most difficult level is recorded as Li(j). After optimizing the space complexity, the formula of the marking function becomes the following form:

$$L_i j = i \qquad test_i j \le test_i (j - t_i) + diff_i$$
$$L_1 j = \begin{cases} 0 & j < diff_1 \\ 1 & j \ge diff_1 \end{cases} \tag{5}$$

The pseudo code of the marking function is shown in Figure 1 below:

```
1.FOR i←1 to n do ←
2.   FOR j ←b to 1 do
3.        IF (test[j-time[i]]+diff[i]) ≤ test[j] do
4.           indexing[i][j]←i
```

Fig.1 Pseudo code of the marking function

The tracking algorithm refers to the tracking of the solution. Only knowing the maximum complexity of the test paper may not be the expected result. When the maximum complexity of the test paper is obtained, the makers want to know which questions are selected into the test paper. This requires the use of a tracking algorithm, that is, starting from the maximum complexity value, moving forward from the back to get a new solution.

The pseudo code of the tracking algorithm is ashown in Figure 2 below:

```
1.y←time_most
2.K←num
3.WHILE indexing[k][y]!=0 do
4.   M←indexing[k][y]
5.   x[m-1]←1
6.   Y←y-time[m]
7.        K=m-1
```

Fig.2 Pseudo code of the tracking algorithm

3. Brute force algorithm and greedy algorithm

Brute force is a direct method of solving problems. The usual way of thinking is directly based on the statement of the problem and the definition of related concepts. If there are n items to choose from, the test paper will have 2n possible combinations of items.

The complexity of the brute force algorithm is calculated as follows:

$$\sum_{i=1}^{2^n}\left[\sum_{j=n}^{1} + \sum_{k=1}^{n}\right] = \sum_{i=1}^{2^n}[\{1+...+1\}(\text{n times})+\{1+...+1\}(\text{n times})]$$
$$= (2n)*[1+...+1](2^n \text{ times}) \quad (6)$$
$$=O(2n*2^n)$$
$$=O(n*2^n)$$

Greedy algorithm is often used to solve optimization problems. Its thinking method means that when solving the problem, it always makes the best choice in the current view. That is, without considering the overall optimality as the premise, the result of this algorithm is a local optimal solution in a certain sense.When analyzing the complexity of the greedy algorithm, first sort each question, the lowest complexity can be O(NlogN), and then calculate the time complexity of the algorithm:

$$\sum_{i=0}^{N}1 = [1+1+1....1] \ (\text{N times}) = N \approx O(N) \quad (7)$$

4. Experiment

Aiming at dynamic programming algorithm, brute force algorithm and greedy algorithm, this paper has done two comparative tests. In the first test, the number of test questions that can be provided in the question bank was increased, while maintaining the maximum possible test time of 50 minutes. In the second test, the maximum time for questioning was increased, and the number of questions in the question bank was fixed at 500. In the two tests, the average time for each question is set to 1-15 minutes, and the difficulty of each question is set to 1-10. The number of questions in the question will start from 10 and gradually increase to 1000. The results of the two tests obtained are shown in Table 1 and Table 2.

Table1. Comparison of results of test one

Number of questions	Result			Best result
	Dynamic programming algorithm	Brute force algorithm	Greedy algorithm	
10	56	56	20	56
25	75	75	21	75
100	156		59	156
300	221		67	221
500	298		94	298
750	315		94	315
1000	340		92	340

Table2. Comparison of results of test two

Maximum time	Result		Best result
	Dynamic programming algorithm	Greedy algorithm	
50	284	88	284
100	409	89	409
200	635	96	635
300	780	79	780
400	913	89	913
500	1080	88	1080

From the data in Table 1 and Table 2, it can be seen that the greedy algorithm is less complex, but it cannot accurately give an optimal result. Dynamic programming and brute force algorithms can often give the best results, but brute force algorithms are too complex to run. When the number of question

banks increases to 100, the actual running time on the computer has exceeded 5 hours, so it is forced .It can be seen that the dynamic programming algorithm has high efficiency in solving the current simple test composition problem.

5. Conclusion

This paper selects the direction of test papers for algorithm analysis. Firstly, by modeling the overall test paper problem, an optimization method for the basic dynamic programming model is proposed, which reduces the space complexity from O(n*b) to O(b). On the basis of optimizing the original model, the overall flexibility of the algorithm is increased by using the marking function and the tracking algorithm. Finally, it has passed many experiments and compared and analyzed with other algorithms, which objectively proves that the algorithm in this paper is better than the other two algorithms have certain practical significance and theoretical value.

Since there are few related researches on the use of dynamic programming for test paper generation, the research in this article can make up for the lack of application of dynamic programming algorithm in test paper generation. It is still a very meaningful work to further expand the application of dynamic programming algorithm in new fields.

References

[1] Y. Sun. Design and implementation of online examination system based on intelligent test paper composition. Beijing: Beijing University of Technology, 2016.
[2] Zhang Mengfan,Wang Changda,et al. Intelligent HSK Self-adaptive Graded Test System[J].Information Technology,2015(09):24(in Chinese).
[3] Xu Zongben,Chen Zhiping, et al.The recent development of the basic theory of genetic algorithm.Advances in Mathematics 2000, 029(002): 193-213.(in Chinese).
[4] Ding Suwei, Yan Hongyin, et al.Research and Application of Intelligent Test Paper Composition Based on Set Random Selection. Computer Development & Applications, 2010: 23(5), 10-11.(in Chinese).
[5] Tang Jing,Shu Xiaosong.Research on the Algorithm of Intelligent Test Paper Generation Based on Backtracking[J].China Computer & Communication,2018(19):69-71.(in Chinese).
[6] Mao C, Xiao L, Yu X, et al. Adapting ant colony optimization to generate test data for software structural testing[J]. Swarm and Evolutionary Computation, 2015, 20: 23-36.
[7] Lin Y, Gong Y J, Zhang J. An adaptive ant colony optimization algorithm for constructing cognitive diagnosis tests[J]. Applied Soft Computing, 2017, 52: 1-13.
[8] WANG S, YI Y. Investigation of Test Paper Auto-generation Based on Improved Adaptive Genetic Algorithm [J]. Science Technology and Engineering, 2006, 4: 468-05.
[9] Jiang Long,Cao Junhao,et al.A Method of Intelligent Test Paper Composition Based on Improved Difference Algorithm[J].Computer & Digital Engineering,2017,45(06):1055-1057+1110.(in Chinese).
[10] He Jianying,Wang Guangqiong,et al.Research on Optimization of Intelligent Test Paper Generation Strategy Based on Genetic Algorithm[J].Computer & Digital Engineering,2019,47(01):130-135.(in Chinese).
[11] Hwang, G. J. . (2003). A test-sheet-generating algorithm FOR multiple assessment requirements. IEEE Transactions on Education, 46(3), 329-337