International Conference on Computer and Information Application (ICCIA 2010)

The Research and Realization of Multi-threaded Intelligent Test Paper Generation Based on Genetic Algorithm

Ying Shan
School of Computer Science and Software Engineering, Tianjin Polytechnic University
Tianjin, P. R. China
E-mail: shanying@tjpu.edu.cn

Abstract—**In this paper, a new multi-threaded intelligent strategy of generating test paper based on genetic algorithm is proposed to replace the current test paper generation algorithms that have the disadvantages of lower power, longer time and difficulty in fulfilling the requirements. This algorithm is compared in a number of experiments with the series genetic algorithm and proved to be faster and better. It can be applied in the situations of practical test paper generation.**

Keywords-genetic algorithm; multi-thread; mathematic model; multi-objective optimization

I. INTRODUCTION

Test paper generation problem is a constrained multi-objective optimization problem. In the process of generating a test paper, the method of generation should take account of form, hardship, differentiation and reliability of examination questions. However, the above factors are restricted to each other. Thus, it is hard for us to solve this problem in classic mathematical method. Among various optimal searching methods, genetic algorithm can meet the multipurpose and optimal demand, because it has the advantage of simplicity, easy-handling, global optimality[1]. For this reason, genetic algorithm is more suitable for test paper generation.

Test paper generation problem based on genetic algorithm give each examination question an independent code. By using three basic operation of genetic algorithm---selecting operation, crossover operation, mutating operation, we can get volume of examination questions which meet the condition of constraint and the complete the test paper generation. To some extent, this method can improve the efficiency of test paper generation algorithm and effectiveness of test paper generation results. For the small-scaled test database, it is effective. But for large-scaled test database, this method will result in long chromosome. It will also lead to nonlinear growth of time spending and inefficiency of operating[2].

Specific to the character of genetic algorithm, to discover a multi-threaded algorithm which is more suitable for test paper generation is the key point of this thesis.

II. MODEL DESCRIPTION

According to the principle of test paper generation, the quality of test paper needs to meet various requirements[3]. In intelligent test paper generation system, examination question should contain the following 7 factors: form, score, difficulty, knowledge point, exposure, cognitive level and chapter. Thus, object matrix of test paper generation is like this:

$$s = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{17} \\ a_{21} & a_{22} & \cdots & a_{27} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{m7} \end{pmatrix} \tag{1}$$

Meanwhile, object matrix should meet the following constrained conditions:

- Constraint of total testing score of all levels in test paper should meet following conditions:

$$Q_t = \sum_{i=1}^{m} C_{4i} \cdot a_{i2} \tag{2}$$

Where t is the number of difficulty levels. Q_t is total score of test paper at t difficulty level. a_{i2} is the score of No. i question. a_{i3} is the difficulty level of No. i question. if $a_{i3}=t$,then C_{4i}=1,else C_{4i}=0 .

- Constraint of knowledge point score should meet following conditions:

$$\begin{cases} L_1 \le \sum_{i=1}^{m} C_{2i} \cdot a_{i2} \le U_1 \\ L_2 \le \sum_{i=1}^{m} C_{3i} \cdot a_{i2} \le U_2 \end{cases} \tag{3}$$

In test paper database, I_1 is important knowledge point, and I_2 is sub-important knowledge point, and their corresponding score range are $[L_1, U_1]$ and $[L_2, U_2]$. if $a_{i4} \in I_1$,then C_{2i}=1,else C_{2i}=0; if $a_{i4} \in I_2$,then C_{3i}=1,else C_{3i}=0, Where a_{i4} is the knowledge point factor of No. i question.

- Constraint of difficulty of examination should meet following conditions:

$$P_t = \sum_{i=1}^{m} C_{1i} a_{i2} \tag{4}$$

Where P_t is expecting score at t difficulty level ($1 \leqslant t \leqslant 5$), if $a_{i3}=t$,then C_{1i}=1,else C_{1i}=0.

- Constraint of test paper total score should meet following conditions:

$$K = \sum_{i=1}^{m} a_{i2} \tag{5}$$

Where K is test paper total score. Default score: 100.

- Constraint of test paper exposure should meet following conditions:

$a_{i5} \leqslant n$, i=1,2,3…m (6)

This constraint defines the maximum appearance time of each examination question. n is decided by teacher.

- Constraint of integral difficulty of test paper should meet following conditions:

$$sumd = \sum_{i=1}^{m} c_{4i} a_{i3} \quad (7)$$

Where *sumd* is designed for constraining integral difficulty of test paper. In other words, the value of *sumd* decides the difficulty of test paper.

Specific to this character, we can model test paper generation by using weighted deviation model. The specific form is like below:

$$\min \sum_{j=1}^{J} w_j (d_{lj} + d_{uj}) \quad (8)$$

Where *j* is number of constraining condition. w_j is weight of No. *j* constraining condition. The more important it is, the higher the weight is. w_j should meet the formula: $\sum_{j=1}^{J} w_j = 1$. L_j and U_j are the upper limit and lower limit of interval of No. *j* constraining condition. d_{lj} is the value which is below L_j. And d_{uj} is the value which is above U_j.

By analyzing the constraining conditions and weighting deviation model of test paper generation, we can get the target function of test paper generation as following:

$$f(X) = w_1 \cdot d_1 + w_2 \cdot (d_{L1} + d_{U1} + d_{L2} + d_{U2}) + w_3 \cdot d_3 + w_4 \cdot d_4 \quad (9)$$

Where *X* is the selected group of examination questions. *S* is the matrix which is composed by attributing index of examination question in *X*. w_2 is the weight of knowledge point score. w_3 is the distribution of different test paper difficulty. w_4 is the weight of test paper integrate difficulty. By searching the answer group which is composed by test paper groups that meet structure constraint, we can get the test paper group which can make the target function *f(X)* be minimum.

$\min f(X)$, $X \in I$ (*I* is solution space) (10)

III. Using Modified Genetic Algorithm to Optimize the Test Paper

It does not have an ideal effect to solve complicated practical problems by using changeless and strategic genetic algorithm. In order to improve the searching ability of genetic operation, this paper will provide methods of multi-thread and punishment system, making it better.

A. Parallel genetic algorithm realized by multi-threads

Each project running on the system is a process, and each process contains one or more thread. Compared with thread, process's operating environment is more independent, However, when the process is dispatched, it will consume two or three times resources as much as thread does, and thread dispatching cost is much lower. Thus, it is a good choice to use multi-thread technology, to improve the effectiveness and to optimize operational results[4].

According to the character of our researching problem, we adopt independent parallelization to optimize genetic algorithm of test paper generation[5]. Each population is evolved independently through one thread. They do not exchange excellent individual in the process of algorithm. The best individual is selected among all populations as the final answer only after the evolution is ended.

The following pseudo code describes the major framework of parallel genetic algorithm:

```
main( )
{
…….
pthread_mutex_t mutex;
pthread_t threads[m];
pthread_mutex_init(&mutex,NULL);
for(i = 0;i < thread_max;i ++){
pthread _create;
}
for(i = 0;i < thread_max;i ++){
ga_solve( )
wait_all_thread_to_finish;
}
Print the result;
……
}
ga_solve( )
{
……
pthread_mutex_lock(mutex);
        if (best_turn.fitness > fbest.fitness) {
                for (i=0;i less than BITS;++i)
                        fbest.cromo[i] = best_turn.cromo[i];
                fbest.fitness = best_turn.fitness;
……
        }
pthread_mutex_unlock(mutex);
…….
}
```

Pthread_create( ) is used to create threads. The data will back to 0 if the creation is successful. Otherwise, it means something wrong. It receives 4 parameters. The first is used to ID backing to thread; the second is used to set up thread's attribute, and the default is NULL; the third parameter is operating function of thread. The fourth are some parameters which will be sent to operating function. If there is only one parameter, it can be delivered directly. If there are many parameters, they should be delivered in the form of structure. In test paper generation, each new thread will operate genetic algorithm separately---the operation of ga_solve() function.

For each thread operating genetic algorithm, it needs to wait for their operating fulfillment. pthread_join() function has two parameters. One is the waiting thread identification, and the other is backing status after transferring. After all threads are operated, the operation of genetic algorithm is ended.

For global variable, we should guarantee that it is operated by only one thread at one time. So here we can use mechanism of thread mutual exclusion. Basic procedures of using mutual exclusion are like these:

- To create and initialize a mutex variable;
- Many threads try to lock mutex variable;
- Only one thread can get the mutex variable, and others are blocked.
- Mutex owner makes some necessary operation, and then owner unlocks mutex.
- Other threads are awakening, continuing to compete for mutex. If one succeeds, repeating above steps.
- Finally, mutex is used and destroyed.

B. Revise of punishment system

For constraining individual in population better, we need to revise the punishment function. Firstly, we should restrict test paper total score of every difficulty levels. When it exceeds certain value, we should punish it. Secondly, we should adopt the same proportion in different level, which means the punishment degree is the same in each level of test paper[6] [7]. Take third-degree-level test paper for instant, its punishment requirement is:

P_3=((vals[2]-51) > 0)?(RATE3*(vals[2] - 51)):-10;

vals[2] is the total score of third-degree-level test paper. RATE3 is the punishment proportion of third-degree-level test paper. If the total score is greater than 51 in third-degree-level test paper, individual will be punished.

Although the constraint to total test paper score of different hardship index is not the same, the punishment requirement is the same. Then by adding P_i value of five different hardship levels, we can get individual fitness formula: cromo->fitness = (P - objt > 0)?(P - objt):0. objt is backing value of objective function. And then individual fitness will become smaller, even to 0. So we can achieve the aim of punishment.

If the individual is excellent, *P* will be greater after being punishment and P-objt's fitness will be better. On the contrary, if the individual is not excellent, P will be smaller after being punished and P-objt's fitness will be worse, even to 0.

IV. SIMULATION EXPERIMENT

In order to confirm the effectiveness and feasibility of test paper generation based on genetic algorithm, this experiment includes 400 questions in test paper database. The distribution of question in each chapter is in TABLE Ⅰ, and the hardship of question in each chapter is in TABLE Ⅱ.

TABLE I. DISTRIBUTION OF QUESTION IN EACH CHAPTER

chapter	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
quantity problem	32	54	41	42	51	68	59	53

TABLE II. HARDSHIP OF QUESTION IN EACH CHAPTER

difficulty coefficient	**1**	**2**	**3**	**4**	**5**
quantity problem	32	84	178	83	23

A. Experiment requirement and specific parameter

The experiment requires generating test paper by using both serial genetic algorithm and parallel genetic algorithm. The total score is 100. The time of examination is 120 minutes. In experiment, parameter has the same setting, like followings: population size is 70; crossover probability is 0.7; and mutation probability is 0.15. The weighting value in target function is as following: the weight of test paper total (w_1) score is 0.5; the weight of important and sub-important knowledge point score (w_2) is 0.2; the weight of test paper hardship in different levels (w_3) is 0.2; the weight of test paper total hardship (w_4) is 0.1.

B. Experimental results

- Comparison of test paper generation

TABLE Ⅲ shows different optimal effect of test paper generation by using basic genetic algorithm and multi-threaded modified genetic algorithm under the circumstance of same population size. Parallel genetic algorithm uses NO.4 thread to test.

TABLE III. THE RESULTS OF SERIAL GENETIC ALGORITHM AND IMPROVED GENETIC ALGORITHM

iteration number	**serial genetic algorithm**					
	best fitness	***Total scores of the questions with the same degree of difficulty***				
		1	***2***	***3***	***4***	***5***
500	60	10	18	51	21	0
800	63	5	18	48	20	9
1000	65	8	7	50	20	15
2000	69	9	8	51	20	12
4000	84	5	26	49	6	14
iteration number	**improved genetic algorithm(4-thread)**					
	best fitness	***Total scores of the questions with the same degree of difficulty***				
		1	***2***	***3***	***4***	***5***
500	72	10	23	32	20	15
800	78	6	21	50	13	10
1000	81	7	20	48	16	9
2000	85	8	18	46	18	10
4000	87	10	23	40	19	8

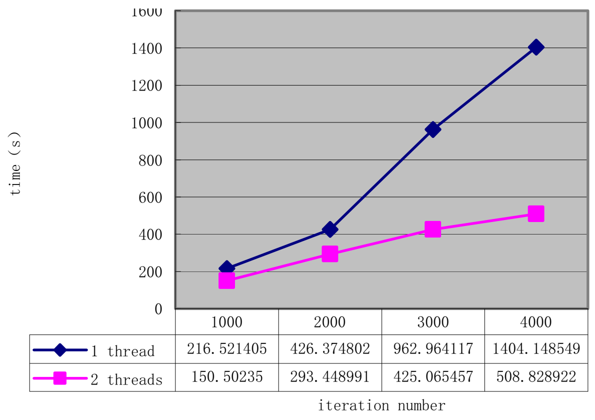
Since the calculated amount of parallel genetic algorithm in No.4 is four times as much as calculated amount of serial genetic algorithm, the operating time of parallel genetic algorithm is two times as much as operating time of serial genetic algorithm at two-core platform.

- Test of speed-up ratio

By using multi-threaded method, we can speeds up the operation of genetic algorithm. When the subpopulation size of each thread of parallel is the same as population size of serial algorithm, under the circumstance of similarity between iterations and serial algorithm, the total calculated amount of parallel algorithm and serial algorithm are same. By comparing the experiments of parallel genetic algorithm and serial algorithm in double threads, we can get the result of optimal effect like TABLE Ⅳshows.

TABLE IV. THE EXPERIMENT RESULTS OF SERIAL ALGORITHM AND MODIFIED ALGORITHM WHEN THE TOTAL EVOLVED ALGEBRA IS SAME

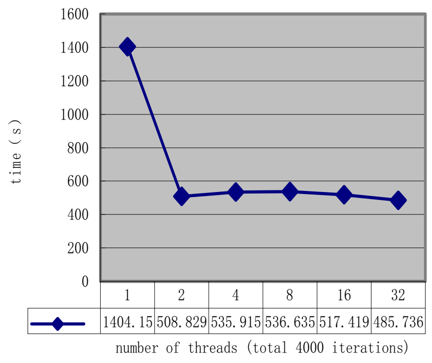
iteration number	**serial genetic algorithm**					
	best fitness	***Total scores of the questions with the same degree of difficulty***				
		1	2	3	4	5
1000	65	4	22	50	16	8
2000	65	8	13	51	20	8
3000	66	8	10	49	19	14
4000	65	1	19	51	19	10
iteration number	**improved genetic algorithm(2-thread)**					
	best fitness	***Total scores of the questions with the same degree of difficulty***				
		1	2	3	4	5
500	65	6	17	50	17	10
1000	65	3	9	49	24	15
1500	66	7	8	51	21	13
2000	65	8	13	49	18	12



	1000	2000	3000	4000
1 thread	216. 521405	426. 374802	962. 964117	1404. 148549
2 threads	150. 50235	293. 448991	425. 065457	508. 828922

Figure 1. Comparison of the acceleration effect

Besides, this experiment tests how the increased thread influences the time when the iteration of parallel genetic algorithm is solid. The testing iteration is 4,000. The test includes speed-up ratio of No.4 thread, No.8 thread, No.16 thread and No.32 thread. The relationship between number of thread and operating time is like Fig. 2.



	1	2	4	8	16	32
◆	1404. 15	508. 829	535. 915	536. 635	517. 419	485. 736

Figure 2. Relationship between the number of threads and the running time

Simulation experiment shows that test paper generation based on modified genetic algorithm can meet the need of examination. It can not only generate test paper quickly with good quality, but have a high success rate.

In conclusion, under the circumstance of that the subpopulation size of parallel genetic algorithm and population size of serial algorithm are same, and the evolved algebras are same, parallel genetic algorithm can have a better effect than serial algorithm. Moreover, under the circumstance of that the total evolved algebra of parallel algorithm is the same as serial algorithm's, parallel algorithm can get the answer with the same quality of serial algorithm, but calculated time of parallel algorithm is much shorter.

REFERENCES

[1] C.Smith Greg, S.Smith Shana, "An enhanced genetic algorithm for automated assembly planning," Roboties And Computer Integrated Manufacturing, vol. 18(5), pp. 355-364, October 2002.

[2] Bingyi Mao, "Research on Database Structure of Composing Test Paper Intelligently System Based on GA" , Computer Engineering and Applications, vol. 39(62), pp .230-232, February 2003.

[3] Lirong Xiong , Jianwei Shi , "Automatic Generating Test Paper System Based on Genetic Algorithm, " 2010 Second International Workshop on Education Technology and Computer Science , March.2010,pp.272-275, doi: 10.1109/ETCS.2010.250

[4] Fun C, Li J B, and Won K P, "Development of a Java-based distributed platform for the implementation of computation intelligence techniques," IEEE:Proceedings of 2004 International Conference on Machine Learning and Cybernetics[C]. Washington:IEEE Press, pp. 4156~4161, August 2004 .

[5] Guo Guangjun, Hu Yuping and Dai Jingguo,"Research and application of multithread-based parallel computing in Java," Journal of Central China Normal University(Natural Sciences),vol. 39(2), pp. 169-173 , June 2005.

[6] Norbert Oster, Francesca Saglietti , "Automatic Test Data Generation by Multi-objective Optimisation," Computer Science, Vol. 4166, 2006,pp. 426-438, doiI: 10.1007/11875567_32

[7] Jones B.F., Sthamer H.-H., and EyresD.E., "Automatic structural testing using genetic algorithms" Software Engineering Journal, pp. 299 – 306, August 2002.