A PERSONALIZED GENETIC ALGORITHM WITH FORGETTING FACTOR FOR INTELLIGENT TEST GENERATION

Wei Wang, Zhendong Niu, Ke Niu, Peipei Gu, Wenjuan Niu, Zhi Huang
Beijing Institute of Technology
Beijing Institute of Technology, Zhongguancun South Street, Haidian district, Beijing, China
vivianiswangwei@qq.com, zniu@bit.edu.cn

ABSTRACT
With the development of computer science and multimedia technology, computer-based testing becomes increasingly popular, especially the intelligent test generation systems. The algorithm used for generating a test paper has a direct impact on the quality and efficiency of intelligent test generating systems. Due to the advantages of parallelism and global space search, the genetic algorithms are recommended for solving the problem of an intelligent test paper composition. However, the traditional genetic algorithm has its own shortcomings, for example, it cannot create a personalized test paper for an individual learner, and it establishes a premature and slow convergence. This paper concerns itself with each user’s current knowledge level and the extent in which a learner forgets. It keeps to the basic principles of Psychology inasmuch as those principles relate to memory and natural memory loss. Utilizing the genetic algorithm, a Personalized Genetic Algorithm with Forgetting Factor (PGAFF) is proposed and used for a multi-constrained test paper composition problem. Experimental results show that the proposed algorithm can support the personalized test generation which can select questions that users haven’t mastered well to composite a test paper. The generated test can help testers find out those questions that they don’t know well and those they may have forgotten. In this view of point, we can see that PAGAFF outperforms existing simple genetic algorithm on the intelligence of test generation.

KEY WORDS
E-learning, personalized genetic algorithms, intelligent test generation system, forgetting curve

1. Introduction
With the development of intelligent tutoring system (ITS) and the computer-aided instruction system (CAI), online testing has become an important method of evaluation. Intelligent Test Generation (ITG) is one of the important issues of online testing. ITG is quick and effective in selecting questions from the test database to compose the test paper according to user requirements.

The ITG system is divided into two parts: the test database and the algorithm. While the efficiency and quality of the composing process depends heavily on the design of the algorithm, the general test generation system has not taken into consideration the user’s mastery degree of questions and knowledge points making the test questions not intelligent or adaptive to users’ individual requirements.

The algorithms used most often by online examination systems are the Stochastic Selection Algorithm [1], the Backtracking Generation Algorithm [2], the Maximum Weight Algorithm [3], and the Genetic Algorithm [4]. The Genetic Algorithm is a random search algorithm referring to the natural selection and genetic mechanisms in biology. It breaks through the limitations of the general Heuristic Neighborhood search algorithms, enabling distributed information gathering and searching on the entire solution space [5]. Genetic algorithms maintain the characteristics of inherent implicit parallelism, fast convergence and better global optimization [6], which easily handle the issues of generating test papers automatically.

However, the Traditional Genetic Algorithm (TGA) is incapable of performing the personalized features of chromosomes. Gu Peipei proposed a personalized genetic algorithm but it didn’t take the forgetting factor into account [13]. In order to realize an intelligent test generation system considering users’ memory retention, this paper will propose a Personalized Genetic Algorithm with Forgetting Factor (PGAFF), which uses the user’s personal information and takes into consideration the user’s master degree and forgetting degree as a preference gene during the crossover process, thereby increasing the proportion of the questions suited for the individual user in the final paper.

2. Background

2.1 Genetic Algorithms

Genetic algorithm (GA) is a search heuristic that mimics the process of natural selection [7]. A highly population is evolved through several generations by selecting two individuals, crossing the two individuals, and mutating characters in the resulting individuals with a given mutation probability.

The basic process of GA is as follows [8]:

a) Initialization: Set the maximum evolution algebra T, set the population size M, set the termination condition of
iteration, and then randomly generate M individuals as the initial population $P_0$.

b) Individual Evaluation: Calculate the fitness of each individual in the population.

c) Selection: Select two parent individuals from a population according to their fitness. Better the fitness, the bigger chance to be selected to be the parent.

d) Crossover: Cross over the parents to generate new individuals according to a crossover probability. If no crossover was performed, copy of the parents as new individuals.

e) Mutation: With a mutation probability, change the value of some spots on the individuals generated in the crossover process to generate a new mutated individual.

f) New population: Place new offspring in the new population.

g) Replace: Use new generated population for a further run of the algorithm.

h) Check: Check whether the termination condition of iteration is satisfied. If it is satisfied, stop and return the best individual in current population.

i) Loop: Go to step 2.

These processes are repeated until all entries in a new generation are filled. Then the old generation is discarded. In a GA which has overlapping generations, only a fraction of the individuals are replaced in each generation. The fitness of each individual depends on the application, and selection is biased towards individuals with higher fitness. Hence the fitness of the overall population is expected to increase in successive generations.

2.2 Ebbinghaus Forgetting Function

Hermann Ebbinghaus was a German psychologist who pioneered the experimental study of memory, and is known for his discovery of the forgetting curve and the spacing effect. In 1885, he discovered the exponential nature of forgetting. The following formula can roughly describe it: $R = e^{-t/s}$. Where $R$ is memory retention, $s$ is the relative strength of memory, and $t$ is time [9].

![Figure 1. Forgetting Curve](image)

Figure 1 is a typical forgetting curve [10]. The forgetting curve illustrates the decline of memory retention in time. The vertical axis represents the remaining amount of memory, and the horizontal axis represents time intervals.

At the beginning, the retention is 100%, since this exactly the point in time when you actually learned the piece of information. As time goes on the retention drops sharply down to around 40% in the first couple of days.

The significance that lies in forgetting function is the meaningful discussion on human thinking simulation and artificial intelligence.

3. Constraints of Item Bank

A personalized Paper Generation problem is a multi-objective decision-making problem. It needs to satisfy multiple constraints, and select a certain number of questions from the huge question library based on the student’s knowledge mastery level to generate an exam paper that meets the requirements. Paper structure is essentially a two-dimensional table. Each row represents a question, and each column represents each attribute value of the question. Suppose there are $n$ test questions in the test database, and each question has five properties, then the database can be represented as a $[n \times 5]$ matrix.

$$
\begin{bmatrix}
Q_1 & K_1 & Diff_1 & Dis_1 & S_1 \\
Q_2 & K_2 & Diff_2 & Dis_2 & S_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
Q_n & K_n & Diff_n & Dis_n & S_n 
\end{bmatrix}
$$

After setting constraint targets for papers, select $m$ questions from $n$ questions to form a paper, so as to meet the needs of users at question type, knowledge, difficulty, discrimination and scores.

The difficulty of the whole paper and the discrimination of the entire paper are calculated as:

$$
\text{diff} = \frac{\sum_{i=1}^{m} (\text{diff}_i \times S_i)}{\sum_{i=1}^{m} S_i} \quad (1)
$$

$$
\text{dis} = \frac{\sum_{i=1}^{m} (\text{dis}_i \times S_i)}{\sum_{i=1}^{m} S_i} \quad (2)
$$

Where $\text{diff}_i$ is the difficulty of question $i$, $S_i$ is the score of question $i$. $m$ is the amount of questions that user requires. $\text{dis}_i$ is the discrimination of question $i$.

4. The Personalized Genetic Algorithm with Forgetting Factor

The personalized genetic algorithm with forgetting factor (PGAFF) considers the user’s personal information, and improves genetic processes. Personalized information represents the user’s mastery degree of questions. The mastery degree is obtained based on the correct rate of the user’s answers and the forgotten extent. The user’s personal information needs to be added into the genetic processes to achieve a personalized genetic inheritance.

The basic flow of PGAFF is shown in Figure 2.
First, encode questions according to the specific encoding rules. Then, choose a certain number of questions from the original question base to generate the initial group of individuals into the mating pool. A fitness value is calculated with formula (10) for each individual. According to roulette wheel selection rules, a crossover probability and a mutation probability, execute corresponding operators, thus generating a new population. Then calculate fitness of the individuals of the new population. When you reach the convergence requirements (meet the fitness requirements or maximum number of iterations), decode and generate a test, otherwise proceed to the crossover and mutation process.

4.1 Personalized Information with Forgetting Rate

When the system is preparing to generate a test paper, not only the properties of questions need to be considered, but also the respondents’ own qualities are required for selecting questions that are in line with the respondents personalized information. PGAFF uses a student’s mastery of the questions as the student’s personalized preference feature information and introduces it into the crossover process. Mastery degree is influenced by two factors: the correct rate and forgetting rate.

4.1.1 The correct rate

The correct rate of a student when he answers the question can reflect the student's understanding level to a certain extent. The correct rate of a student $j$ to a question $i$ can be calculated as formula (3).

$$R_{ji} = \frac{\textit{ui_corrects}}{\textit{ui_tests}}$$

Where $\textit{ui_tests}$ is the total times that a student answers a question and $\textit{ui_corrects}$ is the times that a student correctly answers a question.

4.1.2 The forgetting rate

Student’s personalization features for a test question include not only the mastery degree, but also the memory degree of the question. High correct rate does not mean that the student has mastered the question. Especially for the objective questions, with speculation factors, students may forget these questions after a long time. So it is necessary to repeat the test to review and consolidate.

Table 1 shows the memory retention amount in different time intervals. Suppose the time interval between the last time that user $j$ answered question $i$ and the current time is $r$ hours, and the remained memory rate is $y$. Then $y$ approximately meets $y = 1 - 0.56t^{0.06}$, and the forgetting rate is

$$M_{ji} = 0.56t^{0.06}$$

4.1.3 Mastery degree

A user’s mastery degree of a question describes how much does this user know about this question. Consider users’ mastery degree of questions can be easier to generate tests that are more proper for different user. In PGAFF, users’ mastery degree will appear in the fitness function (formula 10) as the users’ personalized information. If a user can always answer a question correctly, he/she may have a high mastery degree of this question. But as time goes on, he/she may have forgotten about this question, so he may not have mastered it very well. That’s why forgetting factor needs to be considered in PGAFF.

In this paper, mastery degree consists of the correct rate and the forgetting rate. Mastery rate and correct rate is positively correlated, and is negatively related with forgetting rate. The higher is the correct rate, the higher is the mastery degree. The higher is the forgetting rate, the lower is the mastery degree.

The mastery degree of student $j$ to question $i$ is calculated as follows:

$$g_{ji} = \alpha \cdot R_{ji} - \beta \cdot M_{ji}$$

$$= \alpha \frac{\textit{ui_corrects}}{\textit{ui_tests}} - 0.56 \cdot \beta \cdot t^{0.06}$$

Where $\alpha$ and $\beta$ are the weight of correct rate and forgetting rate.
4.2 Chromosome Encoding Design and Initial Population

Using genetic algorithms to solve problems need to map the first solution space to the problem into a code string, that is, the encoding design. Encoding techniques in genetic algorithms (GAs) are problem specific, which transforms the problem solution into chromosomes. Various encoding techniques used in genetic algorithms (GAs) are binary encoding, permutation encoding, value encoding and tree encoding [8].

To improve the efficiency of the algorithm, the value encoding scheme will be used in PGAFF. Value encoding is the technique in which every chromosome is a string of some values. This encoding scheme overcomes the shortcomings of binary coding algorithm. Binary coding occupies a large amount of search space, while there is no space-occupied in the value encoding, so it can shorten the time on decoding.

In this value encoding scheme, a test will be mapped to a chromosome (individual). The number of each question selected in the test is a gene. Separate the chromosome into different segments by different question types. And the genetic operations will be operated within each segment to ensure that the count of questions in each segment remain consistent.

With value encoding, a test with ten questions can be expressed as a chromosome shown in Figure 3.

<table>
<thead>
<tr>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 125</td>
<td>205 300 681 458 509</td>
<td>710 802 923</td>
</tr>
</tbody>
</table>

Figure 3. A chromosome example with value encoding

In this example, the length of this chromosome is ten, which means this test consists of ten questions. Each value is a gene which represents the selected question’s number. Three segments represent three different question types: type 1, type 2 and type 3. So this chromosome indicates that questions [10, 125] of type 1, questions [205, 300, 681, 458, 509] of type 2, and questions [710, 802, 923] are selected to form a test.

4.3 The Fitness Function

In genetic algorithms, the value of fitness function can distinguish the merits of the individual in a group. Fitness function is used to measure the merits of individual performance. The design of fitness function has a significant impact on genetic operations, and is the basis of optimizing individuals [10].

Each question \( Q_i \) has three attributes associated with the constraints, that is, difficulty \( Df_i \), discrimination \( Dt_i \), and personalized information \( Pref_i \), \( 1 \leq i \leq n \).

Assume that the candidate question database contains a collection of \( n \) questions \( < Q_1, Q_2, ..., Q_n > \), and an exam involves \( k \) knowledge points \( < K_1, K_2, ..., K_k > \). One question is corresponding to one knowledge point, but one knowledge point can be corresponding to many questions. The multiple constraints that a paper faces: the number of each kind of questions \(< QT_1, QT_2, ..., QT_t > (1 \leq i \leq t)\), each knowledge point’s value ratio \(< KS_1 , KS_2 , ..., KS_k > (1 \leq j \leq k)\), the expected difficulty degree \( DIFF \) and the expected discrimination degree \( DIS \).

In this paper, the difference between the actual property values and the expected property value are considered in the fitness function. The smaller is the difference, the higher is the fitness of an individual, so the individual is more consistent with the user’s expectations. The objective function in improved personalized test generation consists of four main parts:

1) The average difference between the actual knowledge score distribution and expected knowledge score distribution, \( f_1 \):

\[
f_1 = \frac{\sum_{i=1}^{m} k_{ji} - KS_{ji}}{k}, \quad f_1 \in [0,1]
\]

(6)

2) The average difference of difficulty, \( f_2 \):

\[
f_2 = \frac{\sum_{i=1}^{m} |diff_{ji}| - DIS}{m}, \quad f_2 \in [0,1]
\]

Where \( m \) is the number of questions to be extracted, \( DIFF \) is the expected average difficulty of the whole test paper. \( diff_{ji} \) is the actual difficulty of each test question.

3) The average difference of distinction degree, \( f_3 \):

\[
f_3 = \frac{\sum_{i=1}^{m} dis_{ji} - DIS}{m}, \quad f_3 \in [0,1]
\]

Where \( m \) is the number of questions to be extracted. \( DIS \) is the expected average distinction degree of the whole test paper. \( dis_{ji} \) is the actual distinction degree of each test question.

4) The average difference of mastery degree, \( f_4 \):

\[
f_4 = \frac{\sum_{i=1}^{m} K_{ji} - G}{m}, \quad f_4 \in [0,1]
\]

Where \( g \) is the mastery degree of each question, \( G \) is the expected mastery degree of the entire paper.

\[
g_j (1 \leq i \leq n) \ \text{is the personalized information of student} \ j \ \text{on question} \ Q_i, \ \text{namely the mastery degree on question} \ Q_i, \ 0 \leq g_j \leq 1. \ \text{Specific calculation method is defined in formula (10).}

The objective function is expressed as \( f = f_1 + f_2 + f_3 + f_4, \ f \in [0, 4] \). The optimal case arises when \( f \) is the minimal. Namely:

\[
\text{Min} \ f = \frac{\sum_{j=1}^{k} k_{sj} - KS_{sj}}{k} + \frac{\sum_{i=1}^{m} |diff_{ji}| - DIFF}{m} + \frac{\sum_{i=1}^{m} dis_{ji} - DIS}{m} + \frac{\sum_{i=1}^{m} g_{ji} - G}{m}
\]

(11)
The fitness function $F$ is negatively correlated with $f$:

$$F = 1 - f \quad (12)$$

### 4.4 Genetic Operators

Simple genetic algorithm operators include three basic forms: selection, crossover and mutation. PGAFF also has these three basic operators.

#### 4.4.1 Selection

The principle of selection is that the more adaptable individuals have larger probabilities to contribute for the next generation. The selected individuals will breed the next generation and will produce a new generation of groups by the crossover and mutation operators. Fitness value determines the individual’s chance of being selected. The most frequently used strategies of selection are deterministic selection and roulette wheel selection [11].

In this paper, we used the selection operator implemented in the standard GA, namely the “roulette wheel selection”, which is the simplest form of the proportional selection [12]. Roulette wheel selection, also known as fitness proportionate selection, is a genetic operator used in genetic algorithms for selecting potentially useful solutions for recombination. It is a proportionate selection scheme in which the slots of a roulette wheel are sized according to the fitness of each individual in the population, and an individual is selected by spinning the roulette wheel. When the value of the objective function $f(x)$ is smaller, the fitness is higher, and the chance of being selected is greater.

If there are $M$ individuals in a group and $F(i)$ is the fitness of each individual, the probability of being selected for each individual is as follows:

$$P_i = \frac{F(i)}{\sum_{i=1}^{M} F(i)} \quad (13)$$

#### 4.4.2 Crossover

Crossover is a process of taking more than one parent individuals and producing a child solution from them. Crossover is the most important operator in genetic algorithms. The new individuals have the combined features of their parents. Crossover embodies the idea of information exchange.

Crossover process is executed in each segment as follows:

1) Decode the mother individual and father individual that are generated in the selection process.
2) Set the cross position $cp$ at mother individual $m$ and father individual $d$.
3) Get question $Q_i$ on mom and question $Q_j$ on dad at the position $cp$.

4) Compare $g_{ij}$ the mastery degree of $Q_i$, with $g_{jr}$, the mastery degree of $Q_j$. If $g_{ij} < g_{jr}$, exchange $Q_i$ and $Q_j$.

5) Move the cross position $cp$ forward. If cp arrives at the end of $m$ and $d$, stop. Two new individual child1 and child2 are born now. Then conduct the next step, mutation.

An example of crossover process is shown in figure 4. The $cp$ is on the forth gene. Then compare the user’s personalized information on $cp$, namely the mastery degree of question 300 and question 243, if the $g_{i,300} < g_{i,243}$ exchange them and move $cp$ forward. Otherwise do not exchange questions on $cp$.

<table>
<thead>
<tr>
<th>Child1</th>
<th>10 125</th>
<th>205 300</th>
<th>521 458</th>
<th>622 755</th>
<th>802 923</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mom</td>
<td>10 125</td>
<td>205 300</td>
<td>681 458</td>
<td>509 710</td>
<td>802 923</td>
</tr>
<tr>
<td>Dad</td>
<td>20 18</td>
<td>271 243</td>
<td>521 307</td>
<td>622 755</td>
<td>802 900</td>
</tr>
<tr>
<td>Child2</td>
<td>20 18</td>
<td>271 243</td>
<td>681 307</td>
<td>509 710</td>
<td>761 800</td>
</tr>
</tbody>
</table>

Figure 4. The Process of Crossover

#### 4.4.3 Mutation

This paper uses the single-point mutation strategy, which only execute the mutation operator on certain question types. For a chromosome, randomly generate a real number $r$ within $[0,1]$ as a random probability. If the random probability $r$ is smaller than the mutation probability $P_m$, then execute the mutation operator to the chromosome, otherwise the mutation will not happen.

Mutation process is as follows: Set random mutation position and get question $Q_i$ at the position. Select question $Q_j$ from the remaining unselected questions of the same knowledge and the same question type with $Q_i$. Compare the mastery degree $g_{ij}$ of question $Q_i$ with that of $Q_j$. If $g_{ij} < g_{jr}$, replace $Q_i$ by $Q_j$. Two new children child1’ and child2’ are created after mutation on child1, child2. Check whether child1’ and child2’ are duplicated with existing individuals before adding them to the new generation of the population. If repeated, then discard child1’ and child2’, and redo the mutation process, and even redo the crossover process in necessary.

#### 4.4.4 Termination Condition of Iteration

There are two termination conditions of iteration in this algorithm:

a) The best case is that all the indicators in the test are completely in line with the user’s requirements, namely when the objective function $f=0$ the iteration naturally terminates. In this case, the algorithm finds the optimal solution.
b) Due to the diversity of the user’s requirements, there are many constraints of objective function. Sometimes it may iterate endlessly to accurately satisfy all constraints. Thus a maximum number of iterations is set to stop iteration. The best individual of the last generation is the optimal solution.

5. Experiment & Results

This paper uses an English course database to do a simulation experiment. The course contains 10 units. There are 1000 questions of three types, including 200 dictation questions, 500 multiple choice questions, and 300 spelling questions. Each unit contains 20 dictation questions, 50 multiple choice questions, and 30 spelling questions. Two students are tested and compared. Student A has a certain mastery degree of questions in [0,100], [201,400], [501,600], [701,800], [901,1000], while the rest of the questions have never been tested by A. Student B has certain mastery degree of questions in [101,300], [401,500], [601,700], [801,900], [901,1000], while the rest of the questions have never been tested by B.

Some parameters set in this paper are as follows: initial population size $M=10$; crossover probability $Pc=0.5$; mutation probability $Pm=0.05$; termination condition is the value of objective function $f(x)\leq 0.04$ or the maximum number of iterations $Num=100$; the expected difficulty degree is 0.6; the expected distinction degree is 0.5, the expected value of test knowledge is \{(1,20), (2,10), (5,30), (7,20), (8,20)\}.

Experiments are executed when the expected mastery degrees are set as 0.2, 0.3, 0.4, 0.5. The results are shown in Table 2. A contrast experiment with TGA is doing to be compared to PGAFF.

It can be seen from Table 3 that TGA generates the same test for user A and user B, and the mastery degree of the entire test is imponderable. So it cannot generate personalized test for different users according to their mastery degree of questions. But PGA can generate different tests for different users. And it can generate tests according to the expected mastery degree that user set. And when the expected mastery degree is lower, the fitness is more likely to be low. The system is easy to find out questions that students haven’t mastered well.

With student A’s information, ten tests using PGAFF are conducted. Seeing from Figure 5, we can find that the average grasp values of A are approximately equal to the expected average grasp. But to the same tests, the average grasp values of B vary widely. This can prove that these

<table>
<thead>
<tr>
<th>Test</th>
<th>Expected index (DIFF,DIS,G)</th>
<th>Final questions and mastery degree</th>
<th>Student</th>
<th>Final index (diff,dis,g)</th>
<th>Fitness value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.6,0.5,0.2)</td>
<td>(767:0.578), (401:0.0), (559:0.432), (954:0.617), (126:0.0), (157:0.0), (15:0.308), (62:2.0:0), (447:0.0), (120:0:0)</td>
<td>A</td>
<td>0.614,0.477,0.193</td>
<td>0.015</td>
</tr>
<tr>
<td>2</td>
<td>(0.6,0.5,0.3)</td>
<td>(247:0.648), (210:0.408), (971:0.292), (18:0:0), (331:0:0), (20:0:0), (928:0.227), (80:0:0), (681:0.372), (805:0.408)</td>
<td>B</td>
<td>0.586,0.493,0.235</td>
<td>0.018</td>
</tr>
<tr>
<td>3</td>
<td>(0.6,0.5,0.4)</td>
<td>(525:0.276), (258:0.328), (907:0.361), (780:0.351), (392:0.112), (23:0.099), (788:0.763), (262:0.305), (420:0.295), (438:0.0)</td>
<td>A</td>
<td>0.632,0.477,0.289</td>
<td>0.022</td>
</tr>
<tr>
<td>4</td>
<td>(0.6,0.5,0.5)</td>
<td>(485:0.612), (535:0.0), (582:0.0), (832:0.285), (115:0.226), (838:0.723), (154:0.263), (830:0.107), (403:0.292), (904:0.284)</td>
<td>B</td>
<td>0.553,0.506,0.299</td>
<td>0.018</td>
</tr>
</tbody>
</table>

Table2
The results of PGAFF

300 spelling questions. Each unit contains 20 dictation questions, 50 multiple choice questions, and 30 spelling questions. Two students are tested and compared. Student A has a certain mastery degree of questions in [0,100], [201,400], [501,600], [701,800], [901,1000], while the rest of the questions have never been tested by A. Student B has certain mastery degree of questions in [101,300], [401,500], [601,700], [801,900], [901,1000], while the rest of the questions have never been tested by B.

With student A’s information, ten tests using PGAFF are conducted. Seeing from Figure 5, we can find that the average grasp values of A are approximately equal to the expected average grasp. But to the same tests, the average grasp values of B vary widely. This can prove that these
6. Conclusion

This paper improves simple genetic algorithms, considering forgetting factors, and adds the personalized information to the objective function of genetic algorithm and to the genetic operators. We propose a Personal Genetic Algorithm with Forgetting Factor (PGAFF). Experiment reveals that this algorithm can generate tests for students by selecting questions with higher forgetting degree or lower correct rate, so as to help students consolidate and enhance their memories. The results show that PGAFF is efficient and outperforms traditional genetic algorithms on its personalization.

<table>
<thead>
<tr>
<th>test</th>
<th>final questions: g1&amp;g2</th>
<th>student</th>
<th>Final index (diff,dis.g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(268: 0.365, 0.686), (13: 0.142, 0), (71: 0.326, 0), (381: 0.448, 0), (474: 0, 0.449), (471: 0, 0.219), (741: 0.182, 0), (91:0.223, 0), (485: 0, 0.612), (323: 0.086, 0)</td>
<td>A</td>
<td>(0.56,0.51,0.177)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>(0.56,0.51, 0.197)</td>
</tr>
</tbody>
</table>

| 2    | (284: 0.279,0.638), (315:0.437,0.0), (344: 0.278,0.0), (225: 0.584,0.467), (294: 0.511,0.083), (240: 0.295,0.602), (8: 0.694,0.0), (10: 0.085, 0.0), (723: 0.509,0.0), (884:0.0, 0.253) | A       | (0.50,0.59,0.37)       |
|      |                        | B       | (0.50,0.59, 0.20)       |

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Project No. 61370137), the International Corporation Project of Beijing Institute of Technology (No. 3070012221404) and the 111 Project of Beijing Institute of Technology.

References

[10] [FlashCard Learner: The forgetting curve](http://www.flashcardlearner.com/articles/the-forgetting-curve/)