Heliyon 9 (2023) e17187

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

A test paper generation algorithm based on diseased enhanced genetic algorithm

JunChuan Cui *, Ya Zhou **, Guimin Huang ***

School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, 541004, Guangxi, China

ARTICLE INFO

Keywords:
Distance education and online learning
Improving classroom teaching
Applications in subject areas

ABSTRACT

With the continuous progress of society, tests, and exams appear more and more frequently in people's lives. Faced with the ever-increasing demand for test papers, efficient test paper generation algorithms have become more important. In this paper, we improved and proposed a Diseased Enhanced Genetic Algorithm (DEGA) based on the Genetic Algorithm (GA), and applied it to the test paper generation algorithm. I the crossover operator, the crossover probability that will change in different situations of the population is adopted. According to the characteristics of the test paper generation algorithm, we use the method based on the hamming distance to calculate the distance between individuals in the population. Aiming at the shortcoming that the mutation operator is too random, we designed and used a disease operator that includes three modules: natural disease, infection, and mutation. It effectively guarantees the distance between individuals in the population and improves the shortcoming that GA is easy to fall into a locally optimal solution. Finally, using the College English Test Band 4 (CET-4) questions from 2014 to 2021 as the data set, comparative experiments were carried out on the test paper generation algorithm based on Random Sampling Algorithm (RSA), GA, Enhanced Genetic Algorithm (EGA) and DEGA. The results show that when using the test paper generation algorithm based on DEGA, the generation of test papers is faster, the number of iterations is less, and the algorithm results are significantly better than other algorithms.

1. Introduction

As the most practical method of selecting talents and testing ability, examinations have been widely used in all walks of life in daily society. With the emergence of major online application systems and network teaching systems, the demand for test papers has also become higher, and the test paper generation method becomes more important. The traditional method of generating test papers is often realized by teachers manually selecting appropriate test questions one by one from the test question bank. But the simple method also brings two drawbacks. First, the manual selection of test questions is a cumbersome task that requires a lot of energy from teachers. When the demand for test papers becomes frequent, the supply will often fall short of demand. Secondly, when selecting test questions, teachers can only rely on their own teaching experience to judge whether they are suitable for this test, and this may also lead to poor-quality test papers that cannot meet teaching needs.

* Corresponding author.
** Corresponding author.
*** Corresponding author.
E-mail addresses: 1362368283@qq.com (J. Cui), ccyzhou@guet.edu.cn (Y. Zhou), sendhuang@126.com (G. Huang).

https://doi.org/10.1016/j.heliyon.2023.e17187
Received 6 February 2023; Received in revised form 7 June 2023; Accepted 9 June 2023
Available online 10 June 2023

At present, the common test paper generation algorithms on the market are mainly based on: the random method, the system generates test papers by randomly extracting qualified test questions from the test question bank many times, but the disadvantages are also obvious. If the database is small, the success rate of the test paper will be too low because it is difficult to find suitable test questions, if the database is large, the algorithm will run at a low speed. Depth First Search (DFS)/Breadth First Search (BFS) [1]; based on the random method and adds a backtracking mechanism to speed up the operating efficiency. However, since it is still based on a single initial value to start an iterative solution, when there are too many constraints set by the teacher, there will still be a situation where no solution can be found, so it still has the disadvantages of complex structure and poor global solution ability. Heuristic search algorithms, such as Particle Swarm Optimization (PSO) proposed by Ref. [2]; and GA proposed by Professor [3]; which is based on the natural selection mechanism of Darwin's biological evolution theory, simulating the survival of the fittest in nature. Use the code as an individual gene, and perform operations such as selection, crossover, and mutation on the population to simulate the process of biological evolution, so that the population gradually approaches the optimal solution.

Although the GA algorithm uses a population composed of multiple individuals as the initial solution set to start searching, it has a good global optimization ability, but it still has the disadvantage of being easily trapped in a locally optimal solution. In this paper, based on the GA, we optimized the encoding method and crossover operator, conceived and added a disease operator, and proposed a test paper generation algorithm based on the DEGA. To verify the performance of the algorithm, we use the CET-4 questions from 2014 to 2021 as the data set and use four evaluation indicators such as the success rate, the total iteration numbers, the average distance, and the average fitness. Then we conducted comparative experiments on the test paper generation algorithms based on RSA, GA, EGA, and DEGA. The experimental results show that the algorithm in this paper has a stronger global solution-seeking ability and higher efficiency, which can effectively improve the success rate of test paper generation and reduce the number of iterations.

2. Related work

The test paper generation problem is essentially a multi-constraint optimal solution problem. Its goal is to select appropriate test questions from the test question bank to form the test paper within a reasonable time according to the needs of teaching. Compared with the traditional manual method of test paper generation, the automatic test paper generation algorithm has a fast output speed, and only a few seconds of short calculation can extract a test paper from the database. And the quality of the test paper is high, and it can make the test paper meet the teaching needs by reasonably planning the difficulty and answering time of each test question and other attributes. Considering the advantages of excellent global solution-seeking ability and good robustness, GA is often used to realize test paper generation algorithm, but the ordinary genetic algorithm still has problems such as too many parameters, and the algorithm falls into a locally optimal solution at the later stage of the iteration because the similarity between individuals is too high. Therefore, scholars have conducted various research on the optimization of GA.

The optimization of GA is usually based on the existing genetic operator. Xu, Pei and Zhu [4] designed a random crossover mapping method. When executing the crossover operator, by setting the length of the crossover gene bit, one or two crosspoints may be generated depending on the starting position of the crosspoint to increase population diversity. Behroozi, Hosseini and Sana [5] proposed a teaching–learning-based genetic algorithm (TLBGA). Unlike the mutation operator of the GA, which changes randomly, TLBGA can control the probability of intelligent mutation based on the teacher phase of teaching–learning-based optimization (TLBO), which makes the chromosome close to a better solution, and enhance the convergence speed of the algorithm. Chou, Hsieh and Qiu [6] use a fitness function with statistical measures and fuzzy logic to avoid prematureness and improve global optimization capabilities. Katoch, Chauhan and Kumar [7] introduce the partially matched crossover (PMX) proposed by Ref. [8]. After the execution of the crossover operator to complete the crossover of some genes, traverse the remaining uncrossed parts of the genes, and if duplicate genes are found, replace them with corresponding genes to avoid duplicate genes in the generated offspring individuals.

Some scholars will improve performance by designing new mechanisms in GA. Wang, Zhang, Bai and Mao [20] adopted a Multi-Population Genetic Algorithm (MPGA). Different from the only population of GA, MPGA uses multiple populations for optimization and randomly introduces elite individuals between populations through migration operators to increase the diversity of the population. Based on it [9], optimized genetic operators such as selection and crossover and introduced niche strategies, and proposed an enhanced multi-population niche genetic algorithm (EMPNGA). Deb, Pratap, Agarwal and Meyarivan [10] improved and proposed a non-dominated sorting genetic algorithm (NSGA-II) based on the NSGA. While effectively reducing the computational complexity, it guarantees genetic diversity by stratifying the dominant relationship between individuals in the population and calculating the crowding degree among individuals, and introducing an elite strategy to protect the elite individuals. Sana, Ospina-Mateus, Arrieta and Chedid [11] improve on NSGA-II to address the job rotation problem. After using the crowding comparison operator to eliminate inferior individuals, delete duplicate individuals, and randomly generate new individuals to ensure the diversity of the population. Wang, Cheng, Ersoy, Zhang, Dai and Dong [12,13] adopted the sorting grouping selection (SGA), which divides the sorted population into two groups according to the fitness value, and take a pair of corresponding individuals in each group to complete the crossover operator, to expand the difference between the parent individuals.

Scholars also often use the GA in combination with other algorithms. Xiong and Huang [14] obtained a personalized feature weight matrix through Linear Discriminant Analysis (LDA) training to construct a suitable fitness function and used the mountain-climbing search algorithm (MCA) to simulate the mutation process, replacing genes that can improve fitness until they cannot be found. Zhang, Li and Yin [15] adopted the idea of the greedy strategy, in the iterative process, a greedy mating pool is generated according to the population, and in the selection operation, those individuals with greater prospects are given priority to ensure the convergence speed [16,17]. combined SA with GA, NSGA-II. The excellent local search ability of SA can make up for the shortcomings of GA that are easy to fall into local optimal solutions. Classifying data through NSGA-II and applying SA to a hierarchy of decision rules to accurately

predict the severity of motorcycle traffic accidents. To solve the integrated satellite imaging and data transmission scheduling problem (ISIDTSP) [18], use the greedy strategy in the selection operator, when the number of iterations is small, compare the parents with the children and select the better individuals. It also introduces the idea of population initialization based on a uniform design proposed by Ref. [19]; which can maximize the uniform dispersion of population individuals in the search space.

In this paper, we improve based on GA and propose a DEGA to realize the test paper generation algorithm. Different from the binary coding of the GA, according to the characteristics of the test paper generation problem, we use segmented real number coding and let the test question number be used as the identification of each gene. Then we optimized the existing crossover operator, by using the adaptive crossover operator to make the algorithm automatically increase the crossover probability to speed up the solution when the population diversity is excellent and the individual fitness is low, and using PMX to avoid duplicate genes. To avoid the algorithm from falling into local optimum during the search process, we introduce the niche strategy and propose a disease operator to ensure population diversity. The disease operator simulates the process of organisms infecting diseases from nature and infecting other individuals in the population and causing genetic mutations due to diseases, including three processes of illness, infection, mutation, and recovery. In the process of illness, each individual has a very low probability of getting sick from nature, and the smaller the number of diseased individuals in the population, the lower the fitness of the individual will increase this probability accordingly. In the process of mutation, each sick individual has a high probability of infecting the disease individuals with high genetic similarity. In the process of mutation and recovery, each sick individual will execute the mutation operator once, and then it will be restored to health if the requirements are met.

3. Diseased enhanced genetic algorithm

DEGA has improved many aspects on the basis of GA, such as: using the crossover probability that automatically changes according to the diversity of the population, using disease operations instead of mutation operator, the method of calculating the distance between individuals in the population, and niche strategies. The flowchart is shown in Fig. 1 and the details are as follows.

3.1. Test paper model

In this paper, in order to intuitively represent the characteristics of the test questions, we use a vector containing 6 attributes to represent a test question.

$$\{a, b, c, d, e, f\} \tag{3-1}$$

These 6 attributes respectively refer to (a) test question number (b) score (c) estimated answer time (d) knowledge points under investigation (e) difficulty (f) type. Because the test paper is composed of several test questions, we use a [n × 6] matrix to represent a test paper with n test questions.

$$\begin{bmatrix} a_1 & b_1 & c_1 & d_1 & e_1 & f_1 \\ a_2 & b_2 & c_2 & d_2 & e_2 & f_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_n & b_n & c_n & d_n & e_n & f_n \end{bmatrix} \tag{3-2}$$

3.2. Constraints and objective function

Through the test paper model, the teacher's requirements for the test paper can be easily converted into corresponding constraints, such as:

(a) Total score constraint: used to require the total score of the test paper generated by the algorithm to meet the total score set by the teacher, where *totalScore* is the total score of the target test paper, $\sum_{i=1}^{n} b_i$ means the sum of the score of each question of current test paper, and φ_1 is the deviation between the current test paper and the target test paper in the total score constraint.

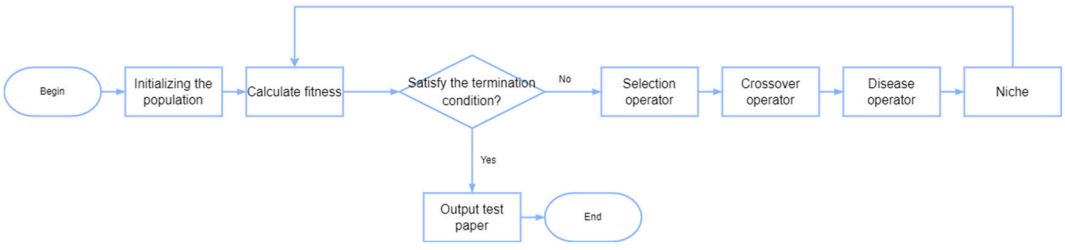

Fig. 1. Diseased enhanced genetic algorithm.

$$\varphi_1 = \left| totalScore - \sum_{i=1}^{n} b_i \right| \tag{3-3}$$

(b) Total time constraint: Similar to the total score constraint, it is used to guarantee the expected answering time of the generated test paper, where *totalTime* is the expected answering time of the target test paper, $\sum_{i=1}^{n} c_i$ means the sum of the time of each question of current test paper, and φ_2 is the deviation between the current test paper and the target test paper in the total time constraint.

$$\varphi_2 = \left| totalTime - \sum_{i=1}^{n} c_i \right| \tag{3-4}$$

(c) Difficulty constraint: used to require the difficulty of the generated test papers. In this paper, we use the weighted average of the difficulty of each test question as the difficulty of the test paper, the test questions with higher scores should have a higher proportion in the test paper, where *difficulty* is the difficulty of the target test paper, $\sum_{i=1}^{n}(b_i * e_i)$ means the sum of the product of the difficulty and the score of each question in the current test paper, and φ_3 is the deviation between the current test paper and the target test paper in the difficulty constraint.

$$\varphi_3 = \left| difficulty - \frac{\sum_{i=1}^{n}(b_i * e_i)}{\sum_{i=1}^{n} b_i} \right| \tag{3-5}$$

The goal of the paper generation algorithm is to make the generated test paper meet the requirements set by the teacher as much as possible, that is, hoped test paper can achieve smaller deviations in each constraint. The objective function $f(x)$ is defined as follows, where m is the number of constraints, each constraint corresponds to a deviation, and ω is the weight corresponding to each deviation, which is set by the teacher based on experience. The larger the weight, the more important the corresponding constraint is for the test paper.

$$f(x) = \sum_{i=1}^{m}(\omega_i * \varphi_i) \tag{3-6}$$

3.3. Encoding

The genetic algorithm usually adopts the binary coding method arranged in 0, 1, but when faced with the problem of generating test papers, the binary coding cannot clearly reflect the structural characteristics of the test paper, which is not conducive to the design of genetic operators. Moreover, when binary encoding is used, the excessively long encoding length will also occupy a huge amount of memory, and a large number of decoding and encoding operations will also affect the algorithm's running speed. Therefore, in this paper, we use segmented real number coding, each gene bit corresponds to a test question, and the number of test questions in the test paper is the length of the gene. Therefore, in this paper, we use segmented real number coding, each gene bit corresponds to a test question, and the number of test questions in the test paper is the length of the gene. According to the different types of test questions, the gene is divided into multiple fragments. The length of each gene fragment is the number of test questions of the corresponding type, and the test question number is used as the identification of the gene. Fig. 2 shows a test paper consisting of 2 questions of type A, 6 questions of type B, and 4 questions of type C, where X_i $i = (1, 2, \cdots, 14)$ represents the question number of the i-th question in the test paper in the database.

3.4. Fitness function

The fitness function is used to simulate the degree of adaptation of organisms to the environment in nature, individuals with higher fitness mean that their genes are better, and they will have better chances of survival and reproduction. When designing a fitness function, the conditions that usually need to be met are good discrimination, the ability to clearly distinguish excellent genes from inferior genes, simple design, and monotonous continuous and non-negative values. Therefore, The fitness function is defined as follows, where $fitness(x)$ is the fitness value of individual x, and $f(x)$ is the value of the objective function of individual x. Based on ensuring that the fitness is a non-negative value, the more the test paper meets the given requirements, the lower its $f(x)$ and the higher the fitness.

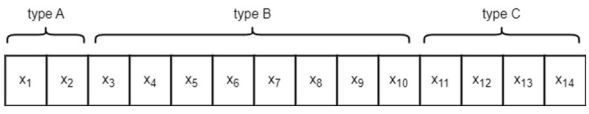

Fig. 2. Encoding.

$$fitness(x) = \begin{cases} 100 - f(x) & if\ f(x) \leq 100 \\ 0 & if\ f(x) > 100 \end{cases} \tag{3-7}$$

3.5. Initializing the population

Because segmented real number coding is used, under the premise of avoiding the selection of the same test question, this paper randomly selects the corresponding type of test questions from the database as gene bits several times to generate initial individuals. Then, according to the number of individuals in the population set in advance, repeat the above operations to generate enough individuals to form the initial population.

3.6. Selection operator

In this paper, we use the roulette wheel selection to implement the selection operation. The selection probability of an individual is proportional to his fitness, the greater the fitness of the individual, the greater the selection probability, the probability of the i-th individual is as follows, where $fitness_i$ is the fitness of the i-th individual and $\sum_{i=1}^{n} fitness_i$ means the sum of individual fitness in the population. And in order to avoid the loss of excellent genes, adopted the optimal individual retention strategy. Check when the selection operation is completed, if the individual with the highest fitness is not selected, automatically find the individual with the lowest fitness in the current population and replace it with the individual with the highest fitness.

$$s_i = \frac{fitness_i}{\sum_{i=1}^{n} fitness_i} \tag{3-8}$$

3.7. Distance between individuals

GA usually uses the Hamming distance to calculate the distance between two individuals, that is, the number of different gene bits at the same position of the genes of the two individuals. However, when faced with the test paper composition algorithm, the following situation may occur: Although two individuals have different test question numbers at a certain gene position, their other attributes of this gene position are the same. At this time, we should have regarded them as the same gene position, but if calculated according to the Hamming distance, the calculated distance will be larger than the actual distance, which cannot play a good role.

Therefore, in this paper, we have adopted a slightly adjusted Hamming distance for the paper generation algorithm. If the two gene bits have the same attributes except for the test question number, they will be regarded as the same question. The distance $d(x, y)$ between individual x and y is as follows, where b is the length of the gene of the individual, x_i is the i-th bit of the gene of the individual x, and $x_{i,?}$ means the attributes corresponding to the i-th bit of the gene of individual x, such as $x_{i,a}$ is the test question number, see chapter 2.1 for other attributes.

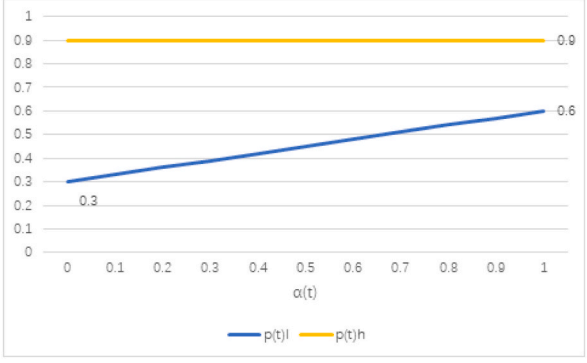

Fig. 3. Function1 (b) From the perspective of each individual, when the individual fitness is lower than the average, we perform the crossover operation with the maximum probability to obtain more possibilities. When the individual fitness is higher than the average, we reduce the probability as the fitness increases to avoid the destruction of excellent genes. The specific formula is as follows, where $f(t)_{max}$ is the maximum fitness value among individuals in the t-th generation population, F is the larger fitness value between the two crossed individuals, and $f(t)_{avg}$ is the average fitness value of the t-th generation population.

$$\begin{cases} d(x,y) = \sum_{i=1}^{b} g(x_i, y_i) \\ g(x_i, y_i) = \begin{cases} 1 & if\ (x_{i,a} \neq y_{i,a})\ and\ (x_{i,b} \neq y_{i,b}\ or\ x_{i,c} \neq y_{i,c}\ or\ x_{i,d} \neq y_{i,d}\ or\ x_{i,e} \neq y_{i,e}\ or\ x_{i,f} \neq y_{i,f}) \\ 0 & otherwise \end{cases} \end{cases} \tag{3-9}$$

3.8. Crossover operator

In this paper, we refer to the adaptive cross-probability strategy adopted by Li, Lei, Wan and Shu (2018), and adjust the formula and parameters based on it. Details are as follows.

(a) From the perspective of the population as a whole, at the early stage of algorithm iteration, the genetic diversity within the population is excellent, and we should increase the crossover probability to improve the speed of the solution. And at the end of the algorithm iteration, the similarity of the genes in the population is high, the crossover operation is often meaningless and we should reduce the crossover probability to avoid redundant operations. The specific formula is as follows, where $[p(t)_l, p(t)_h]$ is the crossover probability range of the t-th generation population, the initial crossover probability range $[p(0)_l, p(0)_h]$ is manually given by the teacher, and the diversity of the t-th generation population $\alpha(t)$ is the average distance between individuals in the population.

$$\begin{cases} p(t)_l = p(0)_l + \dfrac{p(0)_h - p(0)_l}{2} * \alpha(t) \\ p(t)_h = p(0)_h \end{cases} \tag{3-10}$$

Assuming that the initial crossover probability range $[p(0)_l, p(0)_h]$ is [0.3, 0.9], Fig. 3 shows the function image of this formula. It can be seen that when the average distance $\alpha(t)$ is 0, the range of crossover probability $[p(t)_l, p(t)_h]$ is [0.3,0.9], and the crossover probability is the lowest. With the increase of $\alpha(t)$, the population diversity gradually increases, the crossover probability gradually increases and finally reaches the maximum when $\boldsymbol{\alpha}(t)$ is 1, at this time the value of $[p(t)_l, p(t)_h]$ is [0.6, 0.9].

$$p = \begin{cases} p(t)_l + (p(t)_h - p(t)_l) * \sin\left(\dfrac{f(t)_{max} - F}{f(t)_{max} - f(t)_{avg}} * \dfrac{\pi}{2}\right) & if\ F > f(t)_{avg} \\ p(t)_h \quad if\ F \leq f(t)_{avg} \end{cases} \tag{3-11}$$

Assuming that the maximum fitness $f(t)_{max}$ is 90, the average fitness $f(t)_{avg}$ is 60, and the range of crossover probability $[p(t)_l, p(t)_h]$ at this time is [0.4,0.9]. Fig. 4 shows the function image of this formula. It can be seen that when the larger fitness between the two crossed individuals F is low and less than the $f(t)_{avg}$, the crossover probability P remains at a maximum of 0.9, And when F is greater than the $f(t)_{avg}$, as it increases, the P will gradually decrease, and when F reaches the $f(t)_{max}$, P is the minimum value of 0.4.

Because of segmented real number coding, the crossover operator should only be done between each fragment. In this paper, first, we randomly select a point on the gene and use this point and the head of the segment where this point is located as the crossover segment, and then complete the crossover operation by performing PMX on this segment. X_i $(i = 1, 2, \cdots, 8)$ represents the question number of the i-th test question of individual 1 in the database, and Y_i $(i = 1, 2, \cdots, 8)$ represents the question number of the i-th test question of individual 2 in the database. Assuming that X_4 and Y_8 are the same test question, and X_7 and Y_6 are the same test question, the PMX example is shown in Fig. 5.

3.9. Disease operator

As a substitute for the mutation operator in GA, we propose and use a new disease operator to improve the local search ability of the

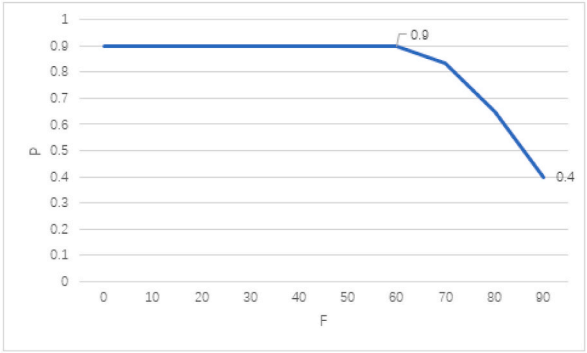

Fig. 4. Function2.

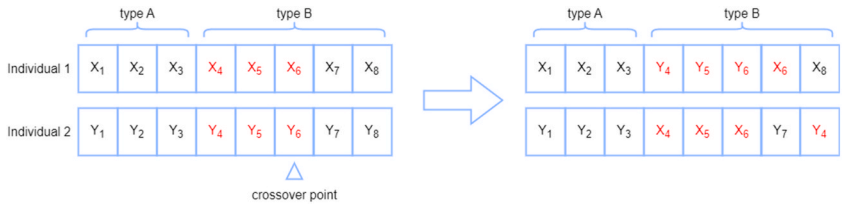

Fig. 5. Partial partially matched crossover.

algorithm. The disease operator simulates the generation, spread, and cure of diseases in nature, and the infected individuals will have genetic mutations due to the disease. Compared with the mutation operator of GA, the disease operator can better ensure the diversity of the population and avoid falling into the local optimal solution. It includes three processes illness, infection, mutation, and recovery, as shown in Fig. 6.

3.9.1. Illness

The illness process is used to simulate the situation that individuals are infected with diseases from nature, which is achieved by making each individual in the population have a small probability of becoming a diseased individual. Similar to the adaptive crossover probability, in this paper, we also consider the probability of illness from the perspectives of the population as a whole and each individual.

$$\begin{cases} q(t)_l = q(0)_l \\ q(t)_h = q(0)_h - \dfrac{q(0)_h - q(0)_l}{2} * \beta(t) \end{cases} \tag{3-12}$$

Assuming that the initial illness probability ranges $[q(0)_l, q(0)_h]$ is [0.1, 0.2], Fig. 7 shows the function image of this formula. It can be seen that when the proportion of infected individuals $\beta(t)$ is 0, the illness probability range $[q(t)_l, q(t)_h]$ is [0.1,0.2], and the illness probability is the highest. With the increase of $\beta(t)$, the number of sick individuals gradually increases, and the probability of illness gradually decreases, and finally reaches the minimum when $\beta(t)$ is 1, at this time the value of $[q(t)_l, q(t)_h]$ is [0.1,0.15].

$$q = \begin{cases} q(t)_l + \left(q(t)_h - q(t)_l\right) * \sin\left(\dfrac{f(t)_{max} - f}{f(t)_{max} - f(t)_{avg}} * \dfrac{\pi}{2}\right) & if\ f > f(t)_{avg} \\ q(t)_h \quad if\ f \le f(t)_{avg} \end{cases} \tag{3-13}$$

The function image of this formula has been described in Fig. 4, so there is no more introduction.

3.9.2. Infection

The infection process is used to simulate the spread of the disease among individuals in the population. Each infected individual has a certain chance of spreading to other healthy individuals, and the more similar genes will lead to an increase in the probability, the specific steps are as follows:

Step1 Calculate the distance $d(x,y)$ between the infected individual x and the healthy individual y.

Step2 If the distance $d(x,y)$ is less than the preset parameter *min distance*, the healthy individual y will have an 80% chance of being infected.

Step3 Repeat step1 and step2 until the entire population is traversed.

3.9.3. Mutation and recovery

The mutation process simulates the genetic mutation caused by disease in nature, which improves the ability of the algorithm to jump out of the local optimum. And the recovery process is used to simulate the recovery of the disease, so as to ensure that the number of sick individuals is within a reasonable range.

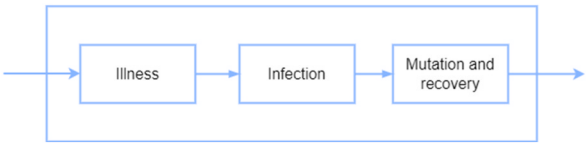

Fig. 6. Disease operator (a) From the perspective of the population as a whole, when there are only a small number of infected individuals in the population, the illness probability needs to be increased appropriately. When the disease has spread and there are many infected individuals in the population, the illness probability needs to be appropriately reduced, so as to ensure that the number of infected individuals in the population is reasonable. The specific formula is as follows, where $[q(t)_l, q(t)_h]$ is the illness probability range of the t-th generation population, the initial illness probability range $[q(0)_l, q(0)_h]$ is manually given by the teacher, and $\beta(t)$ is the proportion of infected individuals in the t-th generation population.

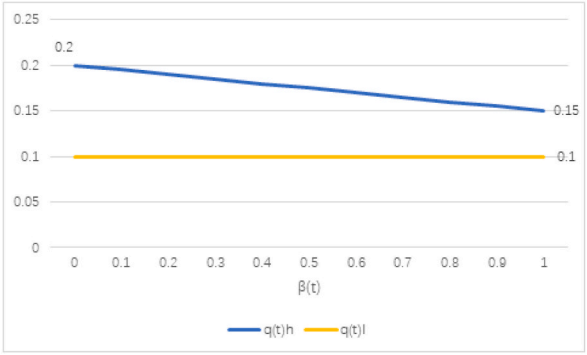

Fig. 7. Function3 (b) From the perspective of each individual, if the individual fitness is lower than the population average, it will get a higher probability of illness to find more genes through the mutation process. If the individual fitness is higher than the average value of the population, the probability of illness will be reduced correspondingly according to the increase of its fitness, so as to avoid the destruction of excellent genes. The specific formula is as follows, where f is the fitness value of the individual.

Step 1 Perform mutation operation on infected individual x to generate new individual x_{new}.

Step 2 Calculate and compare the fitness of individual x and x_{new}, if $fitness(x) > fitness(x_{new})$, cancel the mutation operation, otherwise, save the mutation operation and execute step3.

Step 3 When the new individual x_{new} is retained after mutation, calculate *mdistance* the minimum value of the distance between it and other individuals in the population. If *mdistance* is greater than the preset parameter *mindistance*, the infected individual x_{new} will have an 80% chance of recovery.

Step 4 Repeat the above steps until all sick individuals in the population are traversed.

Because this paper adopts segmented real number coding, the following mutation operator is used: randomly select two different mutation points on the gene segment. Then, under the premise of avoiding repeated genes, replace them with the random genes in the corresponding question bank, an example is shown in Fig. 8, where $X_i\ i = (1, 2, \cdots, 8)$ represents the question number of the i-th question in the test paper in the database, and M_1 and M_2 represent the question numbers randomly generated after the mutation operation.

3.10. Niche strategy

In a large ecosystem, there are often multiple niche populations of similar species. Taking advantage of the competition for limited survival resources in each niche population, the last surviving high-quality individuals are used as representatives of the population, and genetic operators are executed among these representatives to generate a new population. This is the idea of the niche strategy in this paper, which is realized by calculating the distance between every two individuals in the population and imposing penalties on individuals with low fitness when the distance is too close. Penalties will reduce the fitness of the individual and make it easier to be eliminated in the subsequent selection operator to ensure the diversity of the population.

3.11. Termination condition

In this paper, we set two termination conditions for the DEGA algorithm. With the continuous iteration of the population, the fitness of the individual is gradually increasing. When there is an individual that meets the target fitness set by the teacher, the algorithm terminates and the individual is output as the solution. In order to avoid the algorithm may fall into a local optimal solution, a large number of useless iterations will be performed and the efficiency of the algorithm will be affected. Setting parameters: max iteration, when the number of algorithm iterations reaches max iteration but no suitable individual is found, the program will be automatically terminated and it will be regarded as a failure.

4. Experiment

In this paper, we conducted a controlled experiment on the test paper generation algorithms based on DEGA, RSA, EGAA and, GA,

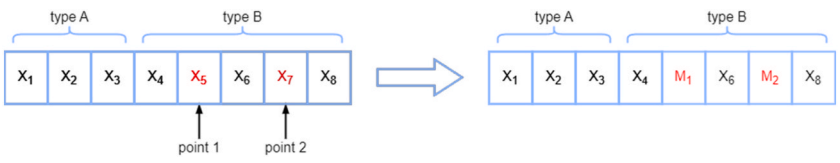

Fig. 8. Mutation.

and verifies the effect of DEGA by comparing the success rate, the total number of iterations, the average distance, and the average fitness.

4.1. Experimental environment

In this paper, we use the previous CET-4 test questions from 2014 to 2021 as the test question bank, which includes 250 listening questions, 150 reading questions, 100 writing questions, and 100 translation questions. The hardware environment and software environment are as follows, and the experimental parameters are shown in Table 1.

Hardware environment.

- Processor: Intel(R) Core(TM) i5-8500 CPU @ 3.00 GHz 3.00 GHz
- Memory: 8G Software environment:
- Eclipse version: 2021-06 (4.20.0)
- Java environment: Java SE-16

4.2. Success rate

The success rate is the most intuitive expression of the performance of the algorithm. A higher success rate means that the algorithm has better results when it is applied to the test paper generation problem. In this paper, we conducted 100 repeated experiments on RSA, GA, DEGA, and EGA, and counted the number of successful test papers to obtain the success rate, as shown in Fig. 9. It can be seen from the figure that the success rate of traditional RSA and GA is low, and the efficiency of generating test papers is very poor, while EGA and DEGA have better results. Among them, DEGA with disease strategy is still significantly improved compared with EGA, and the success rate in the experiment reaches 100%, which proves the excellent performance of DEGA.

4.3. Total iteration times

The total iteration times are related to the efficiency of the algorithm. When the total iterations times of the algorithm are low, it means that this algorithm needs fewer iterations to find the solution and has higher efficiency. In this paper, we conduct 100 repeated experiments on RSA, GA, DEGA, and EGA, and add up the iteration times of each experiment to obtain total iteration times, as shown in Fig. 10. It can be seen from the figure that GA and RSA have a high total number of iterations, requiring more iterations to find a solution, resulting in poor algorithm efficiency. Compared with them, the iteration times of EGA and DEGA are lower, which effectively avoids falling into the local optimal solution and ensures the performance of the algorithm.

4.4. Average distance

The slower the average distance decreases, the better the ability of the algorithm to maintain the diversity of the population, which can effectively prevent the algorithm from falling into a premature situation. In this paper, we conduct experiments on GA, EGA, and DEGA, and record the average distance of the population after each iteration, as shown in Fig. 11. It can be seen from the figure that the decline rate of the average distance between EGA and DEGA is significantly slower than that of GA, and in the early stage of iteration, the effect of DEGA is also slightly better than that of EGA, effectively ensuring the diversity of population genes.

4.5. Average fitness

The average fitness is also used to reflect the efficiency of the algorithm. The faster the average fitness rises, the stronger the algorithm's ability to find solutions and the faster the convergence. In this paper, we conduct experiments on GA, EGA, and DEGA and record the average fitness of the population after each iteration, as shown in Fig. 12. It can be seen from the figure that although the increase rate of the average fitness of DEGA is not much different from that of EGA, it is still significantly improved compared with GA, and has a good convergence speed.

Table 1
Experimental parameters.

Population size	50
Max iteration	50
Target fitness	98
Min distance	0.1
Crossover probability range	[0.3,0.9]
Illness probability range	[0.01,0.2]
Weights	[0.3,0.2,0.4,0.2,1,1,1.2]

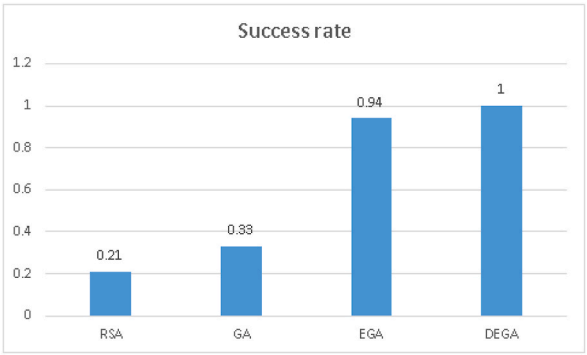

Fig. 9. Success rate.

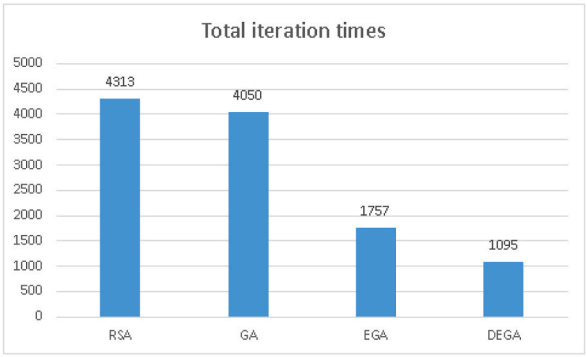

Fig. 10. Total iteration times.

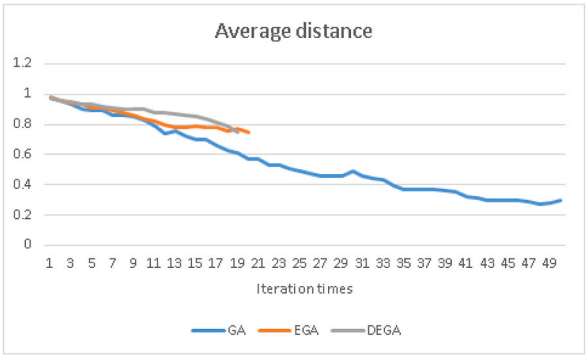

Fig. 11. Average distance.

5. Discussion

In this paper, based on GA, we optimized the traditional crossover operator, population distance, and other parts, added the disease operator, and finally proposed DEGA and realize the test paper generation algorithm. To verify the performance of the algorithm, we conducted a comparative experiment with the test paper algorithm based on RSA, GA, EGA, and DEGA, and the results prove that the solution-seeking ability and convergence speed of the algorithm in this paper is significantly improved compared with the others, which can better complete the task of generating test papers. However, this model is not perfect, it still has certain limitations, and there are many directions for optimization and improvement, we will introduce it in points below.

(1) At present, the number of test questions entered in the database is relatively small. Although it has been able to respond to the needs of the current CET-4 test papers, when faced with more complex requirements, it may be inefficient due to the small test

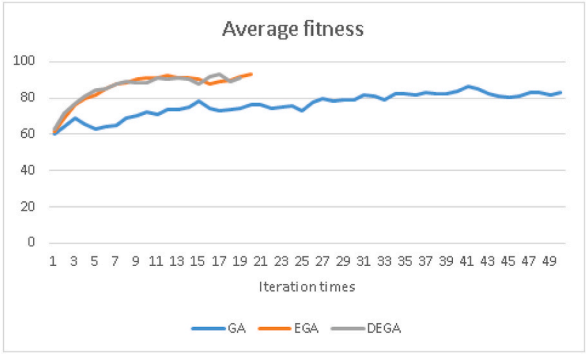

Fig. 12. Average fitness.

question bank. In the future, we plan to continue to collect more excellent English test questions and expand the scale of the test question bank so that the algorithm can achieve better results.

(2) Like other parameters in the GA, the parameters in the disease operator in this paper are currently assigned manually based on experience, and the best results cannot be achieved. By reading relevant literature, the currently conceived solution includes determining the value of each parameter through model training or relevant experimental analysis to ensure the performance of the algorithm.

(3) Compared with the GA, the DEGA needs to calculate the distance between every two individuals in the disease operator, and the complexity of the algorithm is high. Although the number of iterations when generating test papers has decreased significantly, the time spent in each iteration will also increase. To improve efficiency, in the future, I think we can further avoid redundant operations by continuing to optimize the disease operator or reduce the running time through multi-threading.

Author contribution statement

JunChuan Cui: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Wrote the paper. Ya Zhou: Contributed reagents, materials, analysis tools or data. Guimin Huang: Conceived and designed the experiments; Wrote the paper.

Data availability statement

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (1972) 146–160.
[2] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization, Swarm Intelligence 1 (2007) 33–57.
[3] J.H. Holland, Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT press, 1992.
[4] J. Xu, L. Pei, R.z. Zhu, Application of a genetic algorithm with random crossover and dynamic mutation on the travelling salesman problem, Proc. Comput. Sci. 131 (2018) 937–945.
[5] F. Behroozi, S.M.H. Hosseini, S.S. Sana, Teaching–learning-based genetic algorithm (tlbga): an improved solution method for continuous optimization problems, Int. J. Syst. Assur. Eng. Manag. 12 (2021) 1362–1384.
[6] C.H. Chou, S.C. Hsieh, C.J. Qiu, Hybrid genetic algorithm and fuzzy clustering for bankruptcy prediction, Appl. Soft Comput. 56 (2017) 298–316.
[7] S. Katoch, S.S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimed. Tool. Appl. 80 (2021) 8091–8126.
[8] D.E. Goldberg, R. Lingle, Alleles, loci, and the traveling salesman problem, in: Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Psychology Press, 2014, pp. 154–159.
[9] W. Zhang, H. He, S. Zhang, A novel multi-stage hybrid model with enhanced multi-population niche genetic algorithm: an application in credit scoring, Expert Syst. Appl. 121 (2019) 221–232.
[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2002) 182–197.
[11] S.S. Sana, H. Ospina-Mateus, F.G. Arrieta, J.A. Chedid, Application of genetic algorithm to job scheduling under ergonomic constraints in manufacturing industry, J. Ambient Intell. Hum. Comput. 10 (2019) 2063–2090.
[12] J. Wang, Z. Cheng, O.K. Ersoy, P. Zhang, W. Dai, Z. Dong, Improvement analysis and application of real-coded genetic algorithm for solving constrained optimization problems, Math. Probl Eng. 2018 (2018).
[13] F. Wang, G. Xu, M. Wang, An improved genetic algorithm for constrained optimization problems, IEEE Access 11 (2023) 10032–10044.
[14] K. Xiong, X. Huang, Research on auto-generating test paper system based on lda and genetic algorithm, in: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), IEEE, 2018, pp. 416–421.

[15] X. Zhang, X.T. Li, M.H. Yin, An enhanced genetic algorithm for the distributed assembly permutation flowshop scheduling problem, Int. J. Bio-Inspired Comput. 15 (2020) 113–124.
[16] T. Li, G. Lei, F. Wan, Y. Shu, Research on intelligent volume algorithm based on improved genetic annealing algorithm, in: 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), IEEE, 2020, pp. 196–198.
[17] H. Ospina-Mateus, L.A. Quintana Jimenez, F.J. Lopez-Valdes, S. Berrio Garcia, L.H. Barrero, S.S. Sana, Extraction of decision rules using genetic algorithms and simulated annealing for prediction of severity of traffic accidents by motorcyclists, J. Ambient Intell. Hum. Comput. 12 (2021) 10051–10072.
[18] J. Zhang, L. Xing, An improved genetic algorithm for the integrated satellite imaging and data transmission scheduling problem, Comput. Oper. Res. 139 (2022), 105626.
[19] K.T. Fang, D.K. Lin, P. Winker, Y. Zhang, Uniform design: theory and application, Technometrics 42 (2000) 237–248.
[20] D. Wang, Z. Zhang, R. Bai, Y. Mao, A hybrid system with filter approach and multiple population genetic algorithm for feature selection in credit scoring, J. Comput. Appl. Math. 329 (2018) 307–321.