Group-228
Mengying Wang, Yifu Xu
December 7, 2018

Final Project - Genetic Algorithms

Paper Builder

1. Problem

The goal of our project is to build a paper using random questions with an ideal difficulty by Genetic Algorithm. When generating a paper, the user sets various constraints (question types, difficulty, question amounts and so on) according to the requirements, and these constraints are finally converted into the evaluation function. Certain conditions are used to judge the relative proximity to the target of the test paper generated by the program, thereby calculating the relative quality of a test paper.

2. Operators

In our project, it will read all the four-types questions from question files and select questions by random, we used 1 to represent the selected questions, and use 0 to represent the unselected questions, this is the **Genotype**, then we collected these 1 and 0 to generate chromosome, which is the **Expression** in our Genetic Algorithm, and the **Phenotype** is the paper we builed. Then we use some operators to realize **Selected**, **Crossover** and **Mutation**, and defined **Fitness** of paper, those will be described below.

(1) Selected - main.GA.Generate.computeSelectChoice( ) & selectIndividualUseChoice( )

This experiment uses a selection method which is similar to “roulette wheel selection” . The selection process is as follows:

Step 1: Calculate the fitness of the current population, and accumulate the fitness of the current population, and construct a sum representing the total area of wheel

Step 2: Generate a random number between [0,1] called Rand.

Step3: Multiply Rand by the total value Sum. The value between 0 and the total value Sum is obtained, called the runner value, which is the distance of the distance the envisioned runner ball falls into the slot.

Step 4: Accumulate the fitness value of the individual in the population, and check that the individual is selected until an individual has its accumulated value greater than or equal to the runner value.

Each time in the upper generation group, an operation object that selects an individual to perform a genetic operator is installed to generate a new individual and add it as a new population until the new population reaches a specified size.

(2) Crossover - main.GA.Generate.crossover( )

This experiment adopts a method similar to the PMX crossover operator, for example: the chromosome T1=10001-1111100000-10001-0010, T2=01010-1100010101-01100-01010 representing the test paper. The first step: generate the intersection interval randomly, we choose 3, of course, not only this one, this is only a set of intersection positions that meet the requirements. The second step: exchange the two identified intersection positions, and generate new chromosomes T1, T2. T1=10001-1111100000-01100-00101, T2=01010-1100010101-10001-01010.

(3) Mutation - main.GA.Generate.Mutation( )

In order to make sure that the mutation legal, we adopt the chromosomal segment full mutation operator, and the probability of mutation is 0.5%. Its implementation principle is as follows:

Step1: Let chromosome T1=10001-1111100000-10001-00101 first step: determine the variation interval within the range of all individual code strings randomly, select 3 here;

Step 2: Calculate the number of 1 on the determined variation interval of the individual string, Num(1)=2;

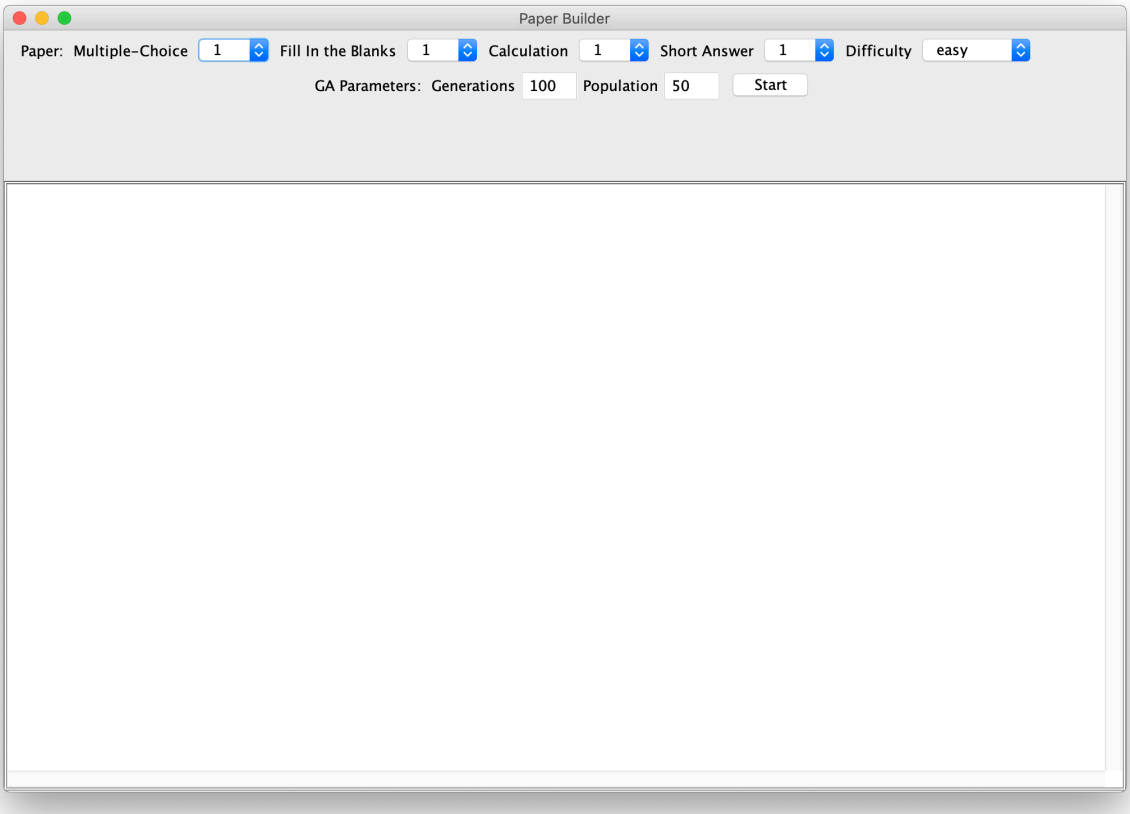
Step3: Apply the mutation operator to the determined variation interval with a predetermined mutation probability Pm. The possible variation result is as follows (here, the mutation occurs): T1=10001-1111100000-00101-00101

(4) Fitness - main.GA.Paper.fitness( )

The difficulty of our questions is 1 ~ 10, and the larger the difficulty we gave, the more difficult the question is. We defined a difficult_array[i][j] to calculate the fitness of each question in the paper, i is determined by the difficulty degree of the paper the user choice (easy, normal or difficult), j is the precise difficulty degree (1 ~ 10) of the question, then we use the sum of all these questions' fitness to represent the paper's fitness.

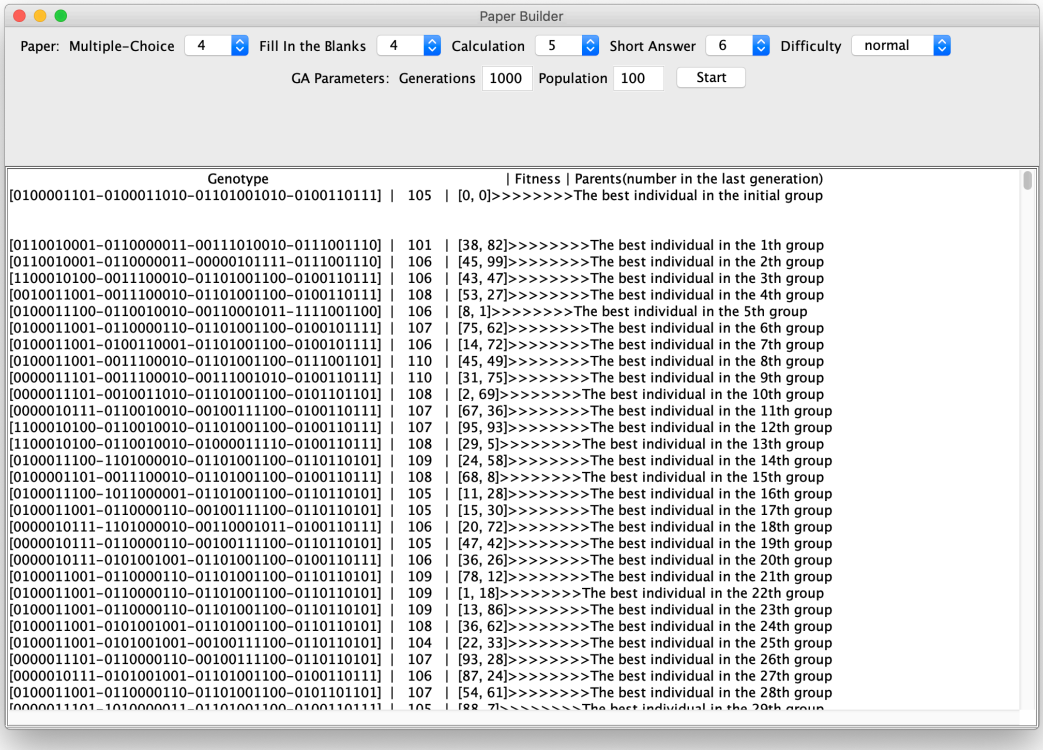
5. Operational Process

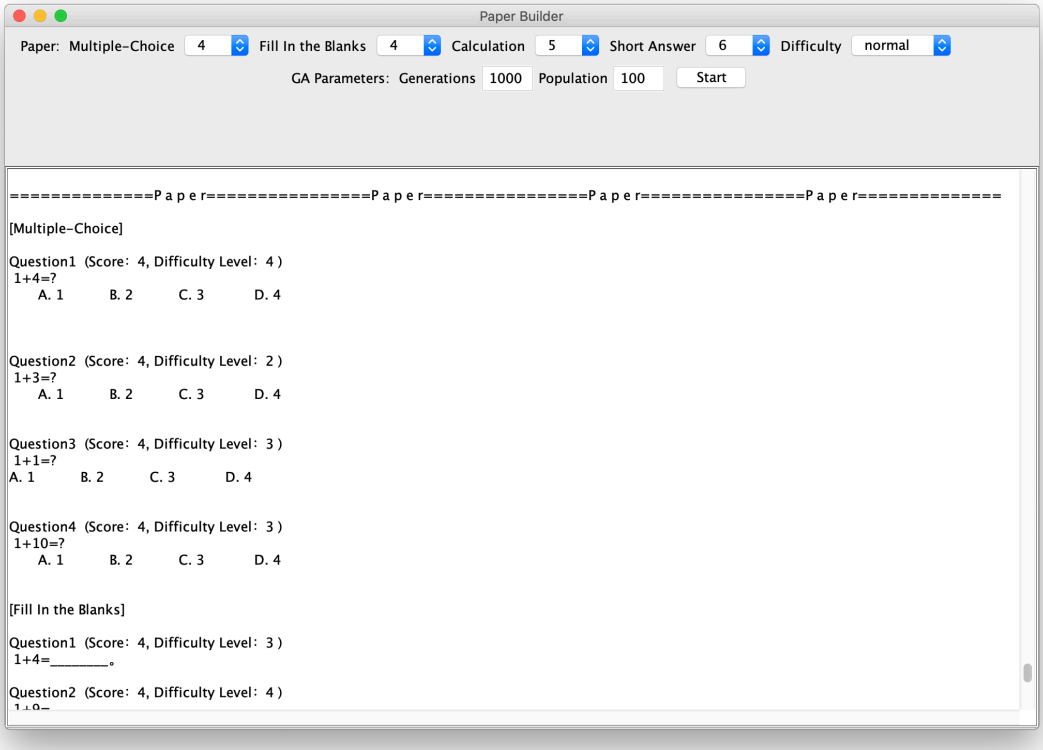
We designed User Interface for our project, When you run the main method in GA.View, it will output like this:



As a user, you can choice the number of every question type and the difficulty of your paper by ComboBox, then you can fill the blank of Generations and Population in the second row, then click "Start" to run Genetic Algorithm and build a paper. You can

see the operational process and the paper built by our system in the output, their is an output example below:





6. Conclusion

Our project can build a random paper by Genetic Algorithm, as you can see in the output, the fitness of every generation changed more and more smoothly during the process, then we can get a paper with the ideal difficulty at last.