

GA Implementation of Auto-Composing Test Paper

Xiaoge Hou , Junhan Ma

1. Problem Description

Our project is to implement Genetic Algorithm on composing exam paper automatically based on problem database. For instance, figure out an efficient way to extract total 40 problems from database whose distribution is 10 multiple choices, 10 true or false, 10 blanks and 10 calculations, additionally its total score is 100 and expecting 0.72 difficulty and 0.8 knowledge points coverage.

2. Algorithm Design ^[1]

Mapping basic genetic algorithm conceptions into our practical problems, we designed a series of classes to implement and solve the problem.

2.1 Gene code: Problem class

In our problem, the problem id would be the gene concept in genetic algorithm because a list of different problems consists of a paper and decides the paper's traits. In our project, the problem entity has fields of id, type, difficulty, score and a set of knowledge points id.

2.2 Gene data bank: ProblemDB class ^[2]

We designed a problem database to store total 5000 gene code. In another word, we hardcode 5000 different problems in database.

- a. Type 1: multiple choice for 1 score from id 1 to 1000;
 - b. Type 2: multiple choice for 2 score from id 1001 to 2000;
 - c. Type 3: true or false for 2 score from id 2001 to 3000;
 - d. Type 4: blank completion whose score is 1 to 4 randomly from id 3001 to 4000;
 - e. Type 5: question and answer whose score is difficulty*10 from id 4001 to 5000;
- All problems' difficulty is randomly generated from 0.3 to 1 and knowledge points count is from 1 to 4, points id is from 1 to 100 (100 kinds of points in total id from 1 to 100).

2.3 Gene expression: Paper class

Paper is the unit which is composed by a set of problem(Genes). Its traits include id, total score, difficulty, KPcoverage(knowledge point coverage) and adaptation degree(fitness). The paper's difficulty is calculated by: $\text{sum}(\text{eachDifficulty} * \text{eachScore}) / \text{totalScore}$. The paper's KPcoverage is calculated by $\text{unitCoverage} / \text{expectCoverage}$, where unitCoverage is convergence of all problems' knowledge points count, expectCoverage is expected points count.

Fitness function in paper class is defined by difficulty and KPcoverage ^{[2][4]}.

$$f = 1 - \left(1 - \frac{M}{N}\right) \times f1 - |EP - P| \times f2$$

* M/N is KPcoverage, EP is difficulty expectation, P is unit difficulty;

* f1 is KPcoverage weight, f2 is difficulty weight, in our project f1 is 0.2 and f2 is 0.8 who are final parameters defined in GlobalWeight class.

2.4 Population: Population class

Population class is the generation of papers, a list of papers. The getBestFitnessPaper() function is going to sort papers by fitness and get the best fitness candidate. Which would be used in selection process in evolution.

The population would be generated by creating a specific number of papers by the given rule including constraints of each problem count and total score. In our project, each generation's population size is always the same.

2.5 Unit standards: Rule class

The rule class defined constraints of total score and each problem type count for generating a paper. And also, the expectation for ideal paper like difficulty and a set of knowledge points hope to cover.

2.6 Evolution mechanism: GA class ^[6]

Flow of Genetic Algorithm for auto-composing test paper:

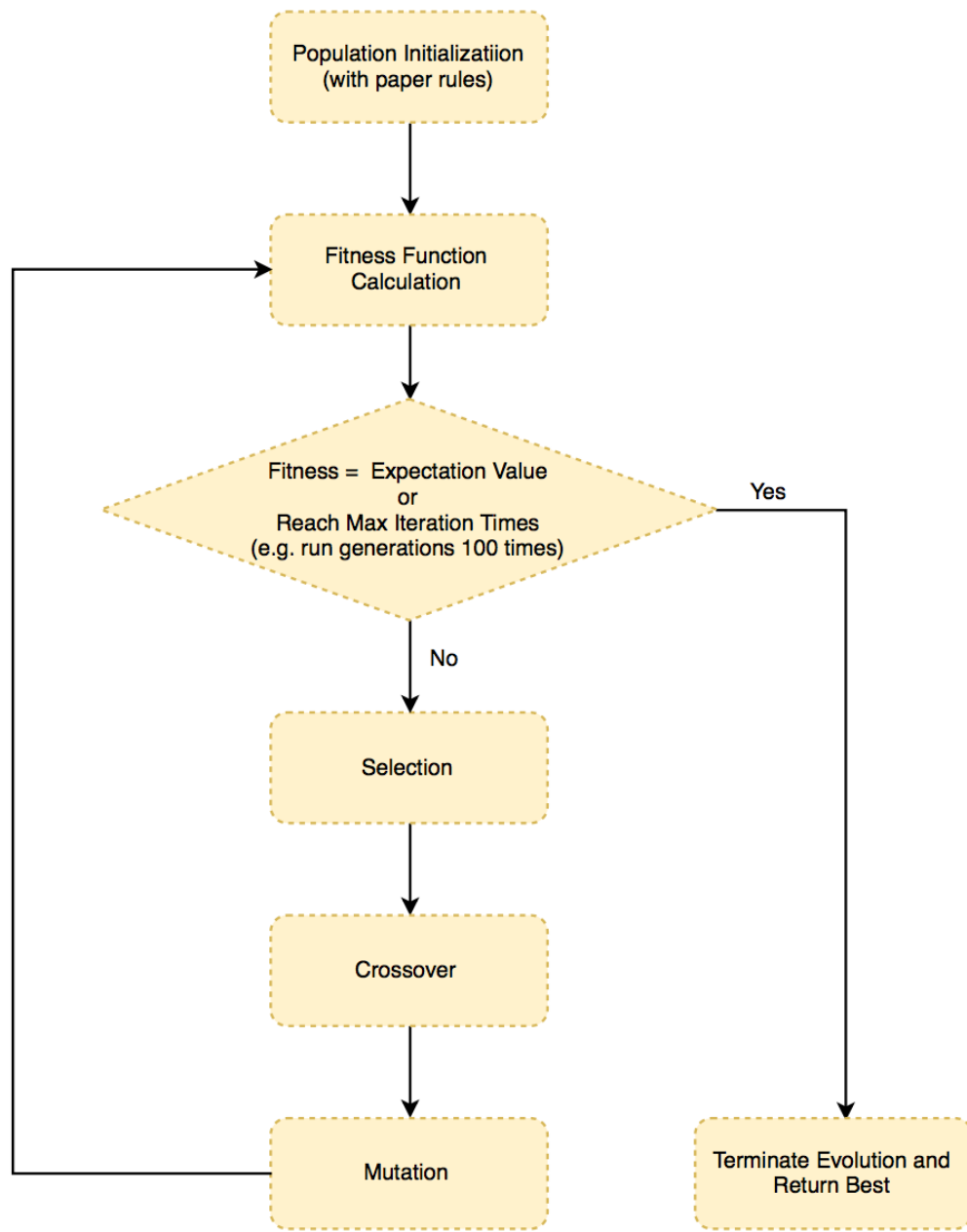


Figure 1. Genetic Algorithm Flow Chart

Evolution methodology: In order to prevent deviation of the evolutionary direction, which is easy to cause local optimal solution, elitism evolution method is used in this project, which means the current fittest member of the population is always propagated to the next generation.

Selection function: Get some good individuals as parents to generate the next generation. In specifically, randomly generate a tournament array, select the best fitness paper in the tournament array as the result of selection, which is the process of simulating the survival of the fittest.

Crossover function: Crossover is a reorganization of chromosomes. We use multi points crossover strategy: randomly generate two integers $n1$, $n2$ between $(0, N)$, where N is the number of genes of one chromosome, and gene segment from two parents between $n1$ and $n2$ or others except segment between $n1$ and $n2$ are reorganized to get a new off-spring. But must ensure that the genes are not duplicated.

Mutation function: Each individual gene has a chance to mutate. If the random probability is less than the mutation probability which is 0.005 in this project, the gene can be mutated. The principle of the mutated gene is that it has the same problem type, the same score, and the same knowledge points as the original gene.

Terminate condition: When there has been no improvement in the population for 100 iterations or the fitness has reached an expectation value, we terminate the evolution and get the best paper.

3. Result

We designed UI to display paper composing process in MainJFrame. Figure 2 is asking for rule of difficulty degree and total grade, additionally asking for parameters of evolution constraints of paper amount in one generation, max evolution times and expected adaptation. Figure 3 is the composing result showed us first generation, evolution process and final result.

Customize Paper

Evolution Result

Auto-Composing Test Paper

Customize Paper

Difficulty Degree:

0.72

Total grade:

100

Evolution Parameters

Paper Amount one Generation:

20

Max Evolution times:

100

Expected Adaptation:

0.98

Compose Paper>>

Figure 2. Settings

Customize Paper

Evolution Result

Evolution Result

Initial Population

Id	KPCoverage	Difficulty	Fitness
1	0.737	0.6	0.704
2	0.684	0.675	0.764
3	0.526	0.67	0.76
4	0.737	0.695	0.78
5	0.579	0.66	0.752
6	0.684	0.649	0.743
7	0.632	0.675	0.764
8	0.526	0.697	0.782

Evolution Process

Generation	Fitness
1	0.918
2	0.931
3	0.933
4	0.949
5	0.957
6	0.957
7	0.957
8	0.957
9	0.962
10	0.962
11	0.962
12	0.971
13	0.971
14	0.972
15	0.98
16	0.98

Expected Difficulty: 0.72

Get the perfect paper in 16 generation

Final Best Paper

Paper ID:

18

KPCoverage:

0.9473684210526315

Difficulty:

0.7082

Fitness:

0.9800336842105264

Figure 3. Result

```

run:
Apr 15, 2018 5:28:22 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Rule(id=1, total=100, difficulty=0.72)
Apr 15, 2018 5:28:22 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 1 Best Adaptation Degree is 0.9179957894736843
Apr 15, 2018 5:28:22 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 2 Best Adaptation Degree is 0.9307284210526315
Apr 15, 2018 5:28:22 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 3 Best Adaptation Degree is 0.9330021052631579
Apr 15, 2018 5:28:22 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 4 Best Adaptation Degree is 0.9492547368421054
Apr 15, 2018 5:28:22 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 5 Best Adaptation Degree is 0.956741052631579
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 6 Best Adaptation Degree is 0.956741052631579
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 7 Best Adaptation Degree is 0.956741052631579
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 8 Best Adaptation Degree is 0.956741052631579
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 9 Best Adaptation Degree is 0.9617010526315791
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 10 Best Adaptation Degree is 0.9617010526315791
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 11 Best Adaptation Degree is 0.9617010526315791
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 12 Best Adaptation Degree is 0.9710273684210529
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 13 Best Adaptation Degree is 0.9710273684210529
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 14 Best Adaptation Degree is 0.9715073684210529
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 15 Best Adaptation Degree is 0.9797936842105265
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Evolution: 16 Best Adaptation Degree is 0.9800336842105264
Apr 15, 2018 5:28:23 PM GeneticAlgorithm.MainJFrame ComposeButtonActionPerformed
INFO: Paper(id=18, totalScore=100, difficulty=0.7082, KPCoverage=0.9473684210526315, adaptationDegree=0.9800336842105264

```

Figure 4. logging information

(Note: Sometimes we cannot get the ideal paper satisfied all constraints which can be solved by increase max evolution iterations or change the mutation rate.)

4. Unit Tests

4.1 Unit Tests Design

We designed unit tests for important functions in projects including followings:

- Population class constructor to generate correct population;
- Get best fitness paper function (mainly sort by fitness function);
- Selection function;
- Mutation function;
- Crossover function;
- Evolution function;
- Choose multi points function in crossover process

4.2 Unit Tests Result

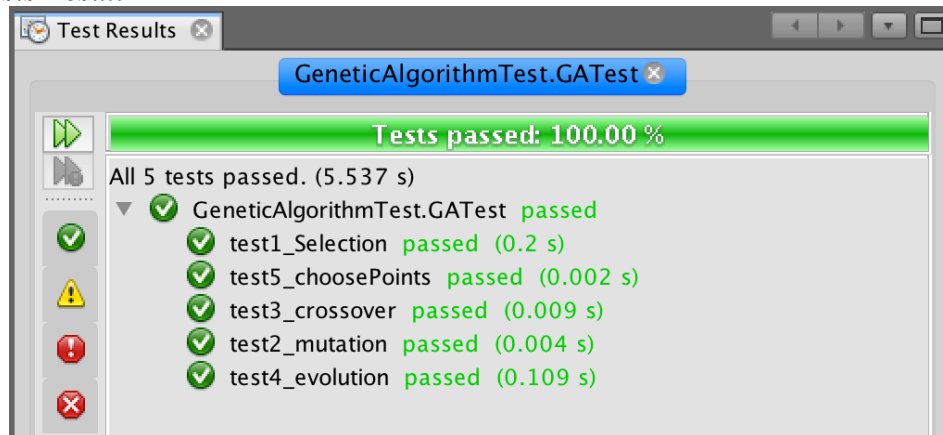


Figure 5. Test Result-part1

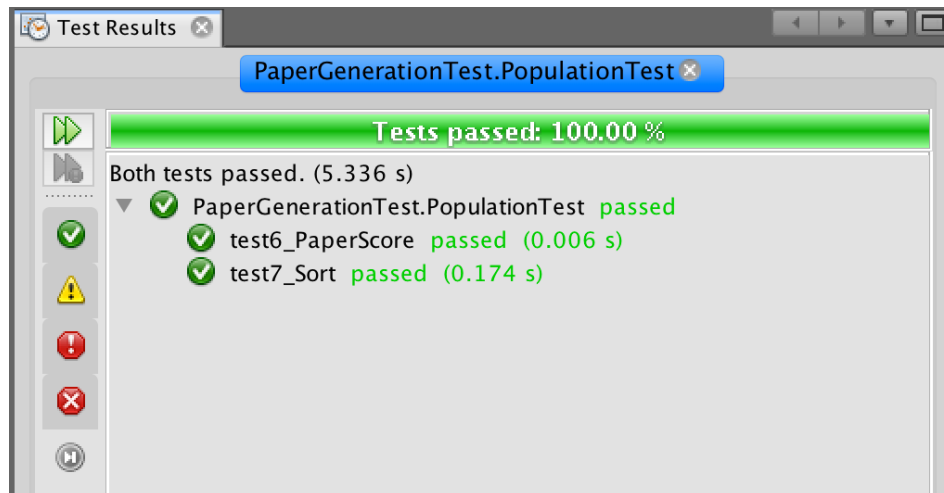


Figure 6. Test Result-part2

5. Conclusion

We take problems in problem database as genes. Each paper's core traits, such like knowledge points coverage, difficulty degree, are expressed by genetic codes, which are also used to design fitness function and calculate adaptation degree, that is a measure of how good a candidate is fit in paper composing requirements which can be customized in user interface.

Also, about our evolution mechanism, we improved our genetic algorithm based on traditional genetic algorithm. We take elitism evolution to prevent deviation of the evolutionary direction and tournament selection to reduce crowding, where we use sorting function to select best fitness paper in a tournament array to propagate. As for mutation probability, mutation probability is best between 0.1-0.001 according to some studies, and higher mutation cause faster convergence, so we finally choose 0.005 as our mutation probability after many trials. It performs good as the figure 3 shows and we also use logging function to keep track of the progress of the evolution and the results display in figure 4.

6. Reference

- [1] <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>
- [2] <http://www.cnblogs.com/artwl/archive/2011/05/20/2052262.html>
- [3] <https://www.jianshu.com/p/7fe9d3bb00ac>
- [4] <https://github.com/jslixiaolin/GADemo>
- [5] <https://ieeexplore.ieee.org/document/5982470/?reload=true&arnumber=5982470>
- [6] https://www.tutorialspoint.com/genetic_algorithms/index.htm