Darwinian Approach to Constraint Satisfaction Problem: Question Paper Generator

Anushree Jagrawal

December 22, 2016

1 Abstract

Tests and examinations play an important role in testing students performance. Traditionally, the question papers are formulated manually by professors. As it is a tedious practice requiring substantial time and effort, professors often generate only a single or a set of few question papers. If students are tested with dissimilar question papers, they gain more knowledge and exposure to variety of problems related to a particular topic. For any educational institute, hence, the task of generating effective question papers is of great importance. This task belongs to the family of Constraint Satisfaction Problems (CSPs). To make this task simpler and efficient, a solution in the form of a Question Paper Generator System using the Genetic Algorithm(GA) with Multi-Objective Constraint Satisfaction and Optimization is proposed.

1.1 Definitions

Constraint Satisfaction Problem A problem composed of a finite set of variables each of which has a finite domain of values and a set of constraints. The goal is to find an assignment of a value for each variable such that the assignments satisfy all the constraints.

Optimization Problem Problem of finding the best solution from all feasible solutions by maximizing or minimizing some function relative to some set, often representing the feasible solutions available.

2 Problem Description

Traditionally, the question papers are formed by professors manually considering various constraints like duration of test, proportion of different types of questions, concepts to be covered in the paper, etc. Being a very tedious practice requiring substantial time and effort, common single paper is generally used for evaluation of students' performance. One of the biggest drawbacks of having identical question papers for all the students is malpractice among students which hampers the vital process of analysing students performance. With an automatic Question Paper Generator System, a lot of time and energy can be saved while maintaining important characteristics of question papers like Proper distribution of concepts, Wide range of topics and Similar difficulty levels.

In this paper, we propose a Generic solution for solving Constrained Satisfaction Problems using Genetic Algorithms. The Constraint Satisfaction Problem taken at hand is to generate question papers satisfying certain parameters. Given a database of N questions (q_1, q_2, q_3, q_N) with attributes as: time Required (t_i), difficulty level(d_i) and type(q_i) (where i [1, N]), we need to select M questions with cumulative values: Total Time Completion(T), average difficulty level (D) and the question type $(Q_t : Qt_1, Qt_2, Qt_3)$, where Qt_1 is proportion of Fill up questions, Qt_2 is proportion of True/False questions and Qt_3 is proportion of select one from multiple choice questions), such that the parameters of selected questions maximize the following:

$$\sum_{i=1}^{M} t_i \leq T,$$

$$\sum_{i=1}^{M} d_i \leq D,$$

$$\sum_{i=1}^{M} qt_i = Qt_1 + Qt_2 + Qt_3$$

3 Literature Review

Various Evolutionary and Optimization Algorithms have been tried on to design Question Paper Generators, e.g. Ant Colony Optimization, Simulated Annealing, Bayesian Classifiers, Semantic Rewriting Approach, Fuzzy Algorithm, etc. Each of the approaches has its own design which works well with certain parameter tuning. However, with Genetic Algorithm at the core much work has been done. However, the Representation and the Fitness function Description vary from differing implementations. Hence, this project is done to understand the application of Genetic Algorithm to generate Question Papers with

a different Fitness Function. For better results, parameter tuning is required in GA as well. Additionally, good combination of Genetic Operator Types also reflects the good results produced by the algorithm.

[5] Yan Li and Ji-Qiang Tang used Genetic Algorithm by mapping the index system to multi-objective functions and optimizing the computing with multi-objective strategy. The index system includes several objects, such as examination paper rubric number object, examination paper rubric type object, examination paper knowledge object, examination paper difficulty distribution object, examination paper rubric reference times object, etc.

[9]Fuzzy Logic based system is proposed by Suraj Kamya, Madhuri Sachdeva, Navdeep Dhaliwa and Sonit Singh in which all parameters where categorized based upon some logic so that the system can be easily acquainted with them. Drawback of this system was that it could only provide results on the basis of analytical and descriptive format; it could not provide Multiple Choice Questions.

[10] Dan Liu , Jianmin Wang and Lijuan Zheng used another Evolutionary Algorithm: Ant Colony Optimization Algorithm where a mathematical model of constraint is built initially according to the requires of papers, and by using the ant colony algorithm, the optimal solution of grouping is obtained. Ant Colony Algorithm has the advantage of finding the solution quuickly.

[3] Mehmet Yildirimh used Complete Representation of Question Paper Candidates. However, crossover and mutation operator of standard GA cannot be directly used for generating test, since integer-coded individuals have to be used and these operators produce duplicated genoms on individuals. In his study, he proposed a mutation operation for preventing the duplications on crossovered individuals.

4 Approach

4.1 Darwins Genetic Algorithm Approach

There are various impressive phenomena going around in nature which have always attracted the interest of researchers due to their admirable perfection. One such phenomenon, that our world has experienced since ages, is the evolution of human beings which is based on the famous Darwins theory of the Survival of the fittest!! This idea has been modelled into an algorithm known as the Genetic Algorithm (GA).

GA is a popular meta-heuristic algorithm efficient in finding the globally optimal solutions among the potentially huge data search spaces and provides solutions to many Multi-Objective Constraint problems. It operates on a population of candidate solutions in a specific problem domain. The structure in the current population is evaluated for its effectiveness as a solution during each generation. Based on this evaluation, a new population of candidate structures is formed using operators like crossover and mutation. This process is iterated until an optimal solution is found or no improvement is achieved after a significant amount of evaluations. The flowchart below describes the algorithm:

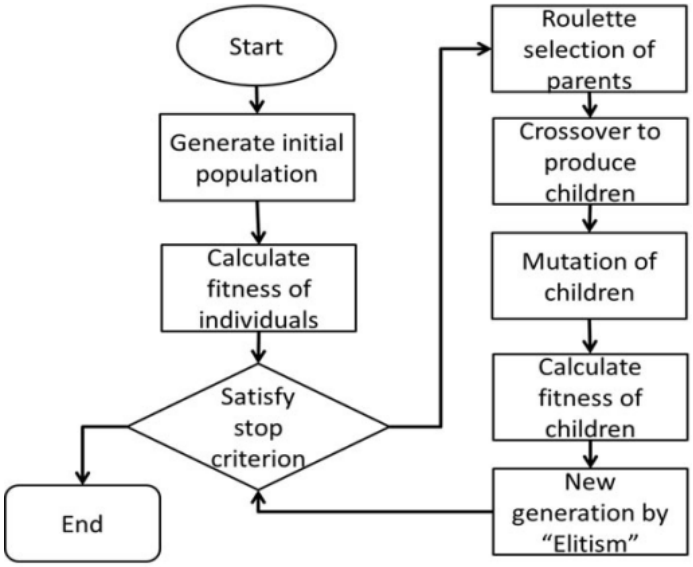

Figure 1: Genetic Algorithm Flowchart

In Genetic Algorithm the transition from one generation to the next consists of four basic operators:

4.1.1 Selection

Selecting individuals for reproduction according to their fitness (objective function value). High fitness components are preferred over low-fitted ones to guide the algorithm to the solution.

4.1.2 Crossover

In reproduction, as it appears in the real world, the genetic material of the two parents is mixed to produce offspring gene sets with some genes of a child coming from one parent while others come from the other parent. This mechanism is called crossover. It is a very powerful tool to maintain diversity of solutions along with the outstanding property that good parents also produce well-performing children or even better ones.

4.1.3 Mutation

Mutation is a random deformation of the solution strings with the positive effect of preserving genetic diversity and, as an effect, avoiding local maxima.

4.1.4 Sampling

Procedure which computes a new generation from the previous one and its offsprings based on their fitness values. On this sampled population next iteration is employed.

4.2 Darwinian Representation for Question Paper Generator System

For a constraint satisfaction problem, candidate solutions play the role of individuals in a population, and the fitness function determines the quality of the solutions. Hence, the efficiency of Genetic Algorithm is determined by the Candidate Representation and the Fitness Function.

4.2.1 Representation

The candidates of the population for Question Paper Generator System are represented as a set of binary strings of fixed length of the form 0100101010101001 where 1 represents the presence of question i in the Candidate and 0 represents the absence of question. As a solution we get a sequence of questions present in the question paper as $q1, q2, q3, qi, qM$ where i denotes the ID of the question present in the Candidate solution. Each q_i has the following parameters:

- t_i (time required to solve the question),
- d_i (difficulty level of the question) and
- m_i (marks of the question)
- qt_i(type of question)

4.2.2 Fitness Function

Fitness function form the core of the Genetic Algorithms. This is the optimization function that is required to be maximized (or minimized) to get the most optimal results for the Constraint Satisfaction problem. For our system, we use the following Fitness Function:

$$F = min \sum_{i=1}^{N} \left(\frac{|T - T'|}{T}\right) + \left(\frac{|D - D'|}{D}\right) + \left(\frac{|M - M'|}{M}\right) + \left(\frac{|Q_t - Q_t'|}{10}\right),$$

where,

$T = TotalTimeCompleteion$, $T' = \sum_{i=1}^{N} t_i$, t_i *is in minutes*

$D = AverageDifficultyLevel$, $D' = \sum_{i=1}^{N} d_i$, $d_i \in [0\text{-}10]$

$M = Total\ Marks$, $M' = \sum_{i=1}^{N} m_i, m_i \in [0{-}5]$

$Qt = Proportion\ of\ different\ question\ types,$

$$Qt' = \sum_{i=1}^{N} qt_i = |Qt_A - qt_A| + |Qt_B - qt_B| + |Qt_C - qt_C|$$

The metaphor between the Genetic Algorithm and the Question Paper Generator System can be understood as follows:

Gene	Questions
Candidate	Question Paper
Population	Set of Question Papers
Fitness of candidate	Value of Fitness Function
Fittest Candidate-	Optimal Solution
Generation of new Population-	Generation of set of improved question papers
Crossover	Creating new and better question papers by modifying good papers

5 Experiment

The proposed approach is implemented using Java Programming Language. The Genetic Operators used are:

- Selection: **Tournament Selection** : At its simplest tournament selection involves randomly picking two individuals from the population and staging a tournament to determine which one gets selected.
- Crossover: **Uniform Crossover** : The uniform crossover uses a fixed mixing ratio between two parents. Unlike single- and two-point crossover, the uniform crossover enables the parent chromosomes to contribute the gene level rather than the segment level. With the mixing ratio is 0.5, the offspring has approximately half of the genes from first parent and the other half from second parent, although cross over points is randomly chosen. Single Point Crossover and Two-Point Crossover were also implemented and tested. However, by running the three types of operators for same number of generations, Uniform Crossover gave best fitness value.
- Mutation: **Random Setting** : With the mutation Probability taken as a parameter, each gene is mutated. A very small mutation rate may lead to genetic drift. A recombination rate that is too high may lead to premature convergence of the genetic algorithm. A mutation rate that is too high may lead to loss of good solutions, unless elitist selection is employed. Hence, mutation rate of 1 per cent is chosen.
- Sampling: **Elitism** : Elitism involves copying a small proportion of the fittest candidates, unchanged, into the next generation. This can sometimes have a dramatic impact on performance by ensuring that the GA does not waste time re-discovering previously discarded partial solutions. Candidate solutions that are preserved unchanged through elitism remain eligible for selection as parents for the next generation. Elitism of 5 per cent is chosen.

Results are generated in the form of question papers as a sequence of Question Numbers included in the paper by applying Genetic Algorithm on the given question bank and ensuring that the constraint satisfaction optimally holds.
The below results were achieved with following parameters:

- Population Size = 20
- Uniform Rate for Crossover = 0.5
- Mutation Rate = 0.01
- Tournament Size = 5
- Elitism = 1
- Termination Condition = No. Of Generations=100

```
Best Candidate:
00100010010011100010100011110110110100000001011101

---------------------------------------OUTPUT---------------------------------------

Question Paper consists of following question Numbers:
2 6 9 12 13 14 18 20 24 25 26 27 29 30 32 33 35 43 45 46 47 49

---------------------------------------ANALYSIS---------------------------------------
Total Time Required:60.0                    Test Time:59.0
Average Difficulty Level Required:0.7       Test Difficluty:0.5181818181818183
Total Marks Required:100.0                  Test Marks:100.0
Type A Ques Proportion Required:20.0%       Test Type A Proportion:18.181818181818183%
Type B Ques Proportion Required:30.0%       Test Type B Proportion:31.818181818181817%
Type C Ques Proportion Required:50.0%       Test Type C Proportion:50.0%
```

Figure 2: Result Set 1

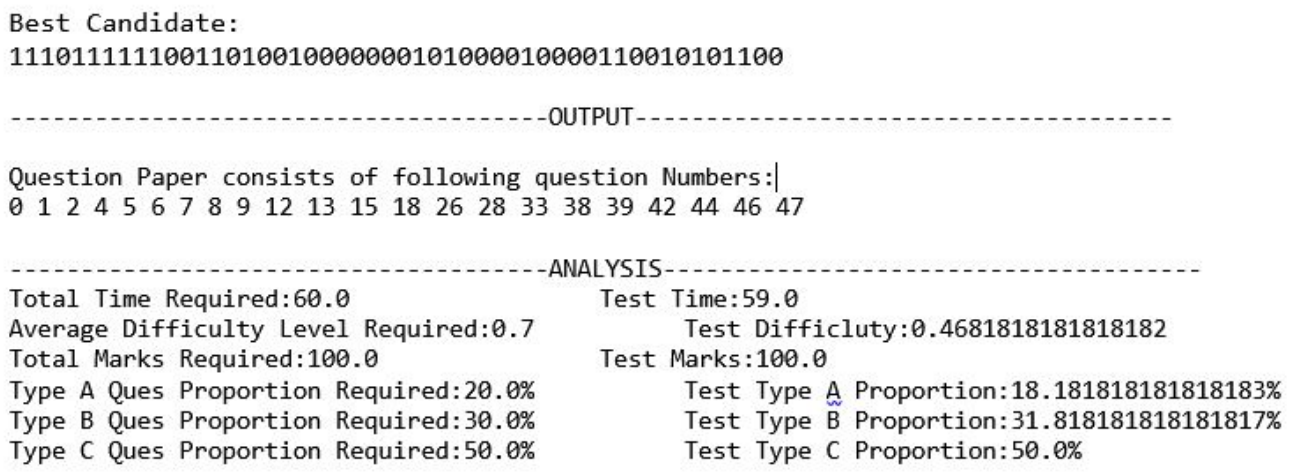

Figure 3: Result Set 2

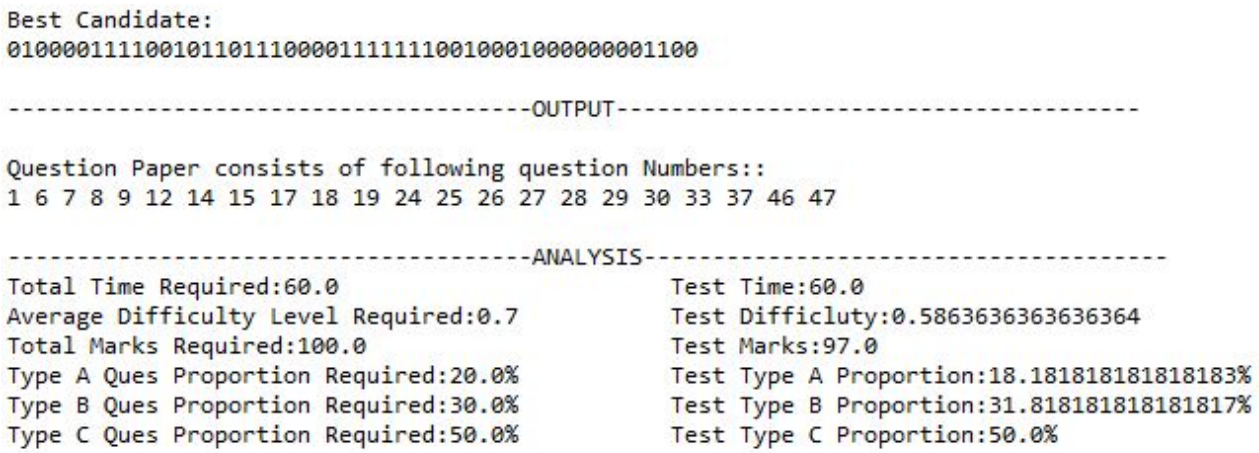

Figure 4: Result Set 3

```
Best Candidate:
01110010010011010111100010100000011101100110100010

--------------------------------------OUTPUT--------------------------------------

Question Paper consists of following question Numbers::
1 2 3 6 9 12 13 15 17 18 19 20 24 26 33 34 35 37 38 41 42 44 48

-------------------------------------ANALYSIS-------------------------------------
Total Time Required:60.0                        Test Time:60.0
Average Difficulty Level Required:0.7           Test Difficluty:0.491304347826087
Total Marks Required:100.0                      Test Marks:99.0
Type A Ques Proportion Required:20.0%           Test Type A Proportion:21.73913043478261%
Type B Ques Proportion Required:30.0%           Test Type B Proportion:30.434782608695656%
Type C Ques Proportion Required:50.0%           Test Type C Proportion:47.82608695652174%
```

Figure 5: Result Set 4

```
Best Candidate:
01100010110000000011100011011110011111100001001000

--------------------------------------OUTPUT--------------------------------------

Question Paper consists of following question Numbers::
1 2 6 8 9 18 19 20 24 25 27 28 29 30 33 34 35 36 37 38 43 46

-------------------------------------ANALYSIS-------------------------------------
Total Time Required:60.0                        Test Time:60.0
Average Difficulty Level Required:0.7           Test Difficluty:0.5454545454545455
Total Marks Required:100.0                      Test Marks:100.0
Type A Ques Proportion Required:20.0%           Test Type A Proportion:22.727272727272727%
Type B Ques Proportion Required:30.0%           Test Type B Proportion:27.27272727272727%
Type C Ques Proportion Required:50.0%           Test Type C Proportion:50.0%
```

Figure 6: Result Set 5

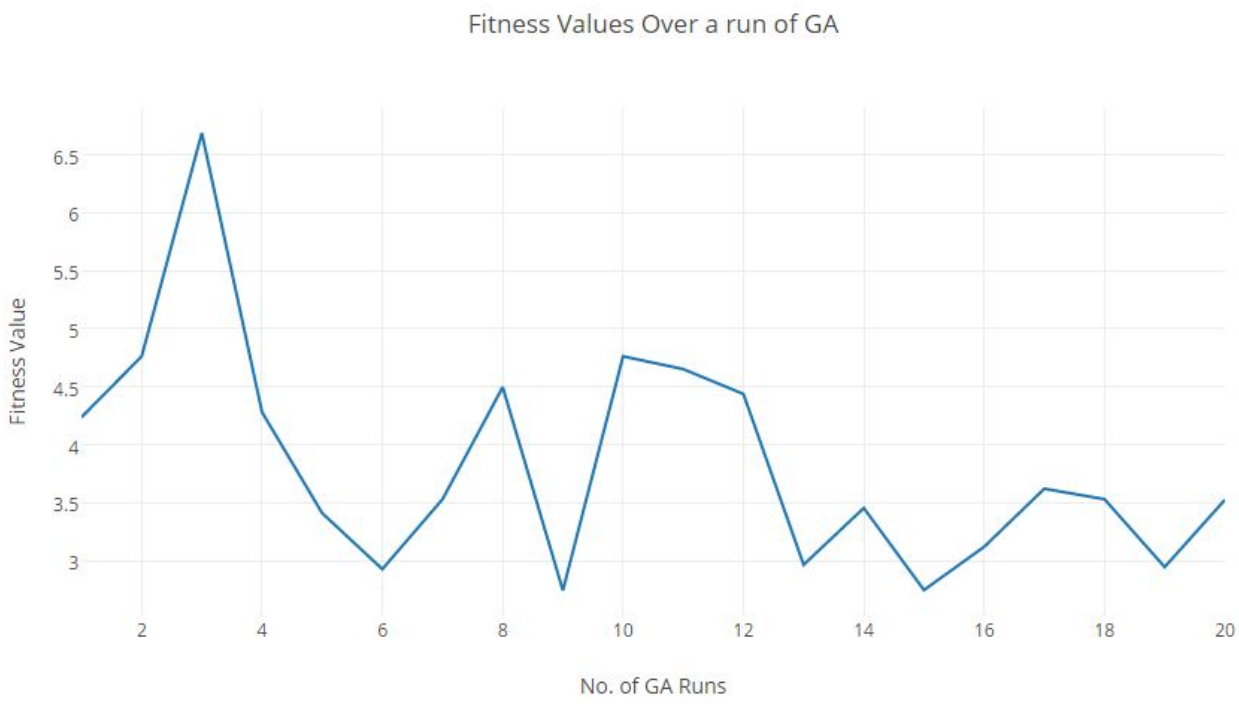

Figure 7: Fitness Value over runs of GA

As a part of result analysis, following graph is generated between the Fitness Value and the runs of the program each running for 100 generations. The diversion between the two runs can be accounted for the Mutation and Crossover Operators which bring diversity to the solution at different rates.

6 Results Analysis

As per the results achieved, it can be seen that Genetic Algorithms work very well on Constraint Satisfaction Problems. It is observed that given multi-objective function as the fitness function, GA tends to find the best solution by maintaining a balance between different constraints satisfaction during multiple runs. With such results , multiple Question Papers with a number of given constraints can be easily generated. Different choices of Genetic Operators also yields different results. For Example: Different Crossover Operators like One-Point Crossover, Multi-Point Crossover, Uniform Crossover were tried. The results obtained were similar with the difference in the number of generations which yielded best results. Uniform Crossover provides more diversity in the crossover process, hence converging to the better solution quickly. One Point and Multi-Point also converged but took more number of generations.

7 Future Work

The problem can be extended to generate multiple question papers together at one run by modifying the stopping condition and elitism procedure. It can further be extended to design Practice and Evaluation System for Students by adding more variety of constraints like Topic wise question distribution and number of Questions. Along with the improvement on algorithm side, an interface can be designed to take the input from the user and output can be generated in the form of PDFs or Documents of Question Papers.

8 References

[1] D. V. Paul, S. B. Naik, P. Rane, and J. D. Pawar, "Use of an Evolutionary Approach for Question Paper Template Generation," in Technology for Education (T4E), 2012 IEEE Fourth International Conference on, 2012, pp. 144-148.

[2] T. Atapattu, "Automated generation of practice questions from semi-structured lecture notes," presented at the Proceedings of the ninth annual international conference on International computing education research, Auckland, New Zealand, 2012.

[3] Yildirim, M. (2010), "A genetic algorithm for generating test from a question bank". Comput. Appl. Eng. Educ., 18: 298305. doi:10.1002/cae.20260

[4] V. M. Kale and A. W. Kiwelekar, "An algorithm for question paper template generation in question paper generation system," in Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on, 2013, pp. 256-261.

[5] http://dx.doi.org/10.4236/jsea.2012.58073

[6] N. Barnier and P. Brisset, "Optimization by hybridization of a genetic algorithm with constraint satisfaction techniques," in Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, 1998, pp. 645-649.

[7] M. Bruce-Lockhart, T. Norvell, and P. Crescenzi, "Adding Test Generation to the Teaching Machine," Trans. Comput. Educ., vol. 9, pp. 1-14, 2009.

[8] R. Kowalczyk, "Using constraint satisfaction in genetic algorithms," in Intelligent Information Systems, 1996., Australian and New Zealand Conference on, 1996, pp. 272-275.

[9] K. Suraj, S. Madhuri, D. Navdeep and S. Sonit, Fuzzy Logic Based Intelligent Question Paper Generator IEEE International Advance Computing Conference (IACC), 2014.

[10] L. Dan, W. Jianmin, Z. Lijuan, "Automatic Test Paper Generation Based on Ant Colony Algorithm" JOURNAL OF SOFTWARE, VOL. 8, NO. 10, OCTOBER 2013.