Assisted Genetic Algorithm To Schedule University Examinations

Saad Bazaz i180621
Abdul Rehman Subhani i180732

Bachelors of Computer Science, National University of Computing and Emerging Sciences, Islamabad, Pakistan.

Submitted to: Dr Labiba Fahad

ABSTRACT

Our objective was to run a genetic algorithm which creates a schedule for a series of university examinations. It was to pass some basic and hard constraints, and any 3 soft constraints. We used a self-constructed basic dataset to test our implementations, then applied some optimization techniques before using the final dataset.

Keywords –Evolutionary Technique, Genetic Algorithm (GA), Timetable Scheduling.

Date of Submission: 06-05-2021

I. INTRODUCTION

As part of our Artificial Intelligence Project, we were instructed to build a Genetic Algorithm to create a schedule for a University Examination schedule. We were to run an unassisted Genetic Algorithm to handle it, but after multiple unsuccessful and time-consuming attempts, we decided to add a simple heuristic.

Being inspired by spirituality and the Quranic concept of human inception and evolution, we thought it interesting to add some similar traits.

II. PROBLEM DESCRIPTION

2.2 Hard Constraints

• An exam will be scheduled for each course.
• A student is enrolled in at least 3 courses. A student cannot give more than 1 exam at a time.
• Exam will not be held on weekends.
• Each exam must be held between 9 am and 5 pm
• Each exam must be of 3 hours duration
• Each exam must be invigilated by a teacher. A teacher cannot invigilate two exams at the same time.
• A teacher cannot invigilate two exams in a row.

2.2 Soft Constraints

• All students and teachers shall be given a break on Friday from 1-2.
• A student shall not give more than 1 exam consecutively.
• If a student is enrolled in a MG course and a CS course, it is preferred that their MG course exam be held before their CS course exam.

• Two hours of break in the week such that at least half the faculty is free in one slot and the rest of the faculty is free in the other slot so the faculty meetings shall be held in parts as they are now.

III. INPUTS AND OUTPUTS

3.1 Inputs

We were given CSV files with the data needed to test against the algorithm.

The following are some modifications made to the files:

• Each student, instructor and day was given a unique identifier (digit) to speed up the matching process. (String matching was taking up a lot of computational time)

We also created an additional array containing names of Classrooms.

3.1 Outputs

Fitness value of the best solution, along with entries of each time-slot.

IV. GENETIC ALGORITHM APPROACH

4.1 Data Structures

Population: The population consists of an array of selected chromosomes. Each chromosome is a fully prepared timetable.

[Chromosome 1, Chromosome 2, Chromosome 3, Chromosome 4, ...]

Chromosome: Chromosome consists of an array of genes.

[Gene 1, Gene 2, Gene 3, Gene 4, Gene 5, ...]

Gene: Gene is a complete entry in a timetable. It contains:

Day of exam,
Start time of exam,
End time of exam,
Invigilator (a teacher),
A list of students taking the exam,
A classroom

4.2 Fitness Calculation

We try to maximize the quality of the chromosome by assigning points upon every constraint passed. For hard constraints, the points assigned are fixed.

However, for soft constraints, we have assigned variable points (less than the hard constraints, though) based on their priority.

4.3 Heuristic Function

Whenever a constraint's passing points are less than a desired amount for a specific gene, we add only the relevant fields to a list. This list is provided to the mutation function, which uses it to only modify the fields which actually need to be modified.

4.4 Evaluation

Here we use some Quranic concepts. We select the best two "representatives" of the "nation". If they are better than the previous ones, then there is no punishment. But in the case that there are no better representatives than the last *n* representatives, we first send some "heavenly assistance" (that is, we add the "previous best solution" to the chromosome pool).

If even then the representatives do not improve, we inflict godly punishment upon the population by completely eradicating them and replacing

them with newer people. We also change the "fundamentals" of the Universe (in our case, we increase the number of days).

4.5 Crossover

We use random fixed point value crossover technique.

4.6 Mutation

Mutation occurs based on a random probability. Mutation can happen on 1) chromosome level, or 2) gene level.

In gene level, we either use the heuristic or generate a completely new Gene.

In chromosome level, we either delete a Gene or generate a completely new Chromosome.

4.7 Selection

We use roulette wheel selection to select two representatives from the population. We have also implemented elitism, which would make more sense in our case.

4.8 Optimizations

To increase algorithm speed, we made the following optimizations (but the constraints are still being checked):

• Instead of randomly selecting a time, we select a start time between *(lower limit)* and *(upper limit - 3)*, and fix the end time as *(start time + 3)*. This ensures that each exam is of 3 hours and is also between the allowed times.

• When initializing students in a Gene, we select those having the same course as already randomly selected in the Gene.

• We exclude Saturday and Sunday from the list of days.

REFERENCES

[1] University Timetable Scheduling Using Genetic Algorithm Approach Case Study, http://www.ijera.com/papers/vol8no12/p2/E0812023035.pdf

[2] Classroom Slides, https://classroom.google.com/u/1/c/MjkzNzAzNjkxODQw/m/MzIxMTEyNzQzNjYz/details