AUTOMATED COLLEGE TIMETABLE GENERATOR

[1] K.Ajith Kumar, [2] S.Raja Durai, [3] Dr.J.V.Anchithalagammai
[1], [2] UG Scholar-Department of CSE,Velammal College of Engineering and Technology, Madurai, Tamilnadu,
[3] Assistant Professor- Department of CSE, Velammal College of Engineering andTechnology, Madurai, Tamilnadu.

Abstract: *Most colleges have a number of different courses and each course has a number of subjects. Now there are limited faculties, each faculty teaching more than one subjects. So now the time table needed to schedule the faculty at provided time slots in such a way that their timings do not overlap and the time table schedule makes best use of all faculty subject demands. We use a genetic algorithm for this purpose. In our Timetable Generation algorithm we propose to utilize a timetable object. This object comprises of Classroom objects and the timetable for every them likewise a fitness score for the timetable. Fitness score relates to the quantity of crashes the timetable has regarding alternate calendars for different classes. Classroom object comprises of week objects. Week objects comprise of Days, Days comprises of Timeslots. Timeslot has an address in which a subject, student gathering going to the address and educator showing the subject is related. Also, further on discussing the imperatives, We have utilized composite configuration design, which make it well extendable to include or uproot as numerous obligations. In every obligation class the condition as determined in our inquiry is now checked between two timetable objects. On the off chance that condition is fulfilled i.e. there is a crash is available then the score is augmented by one.*

INTRODUCTION

Time Table Scheduling Using Genetic Algorithm Time Table Scheduling is an **NP-hard** problem and hence polynomial time verifiable using genetic algorithms. It a typical scheduling problem that appears to be a tedious job in every academic institute once or twice a year. In earlier days, time table scheduling was done manually with a single person or some group involved in task of scheduling it manually, which takes a lot of effort and time. Planning timetables is one of the most complex and errorprone applications. Timetabling is the task of creating a timetable while satisfying some constraints. There are basically two types of constraints, soft constraints and hard constraints.
Soft constraints are those if we violate them in scheduling, the output is still valid, but hard constraints are those which if we violate them; the timetable is no longer valid. The search space of a timetabling problem is too vast, many solutions exist in the search space and few of them are not feasible. Feasible solutions here mean those which do not violate hard constraints and as well try to satisfy soft constraints. We need to choose the most appropriate one from feasible solutions. Most appropriate ones here mean those which do not violate soft constraints to a greater extent. In this project hard-constraints have been taken care of strictly and it has been ensured that soft-constraints are as well followed as much as possible.

Soft-constraints (flexible):

- More or less equal load is given to all faculties
- Required time (hours per week) given to every Batch

Hard-constraints (rigid):

- There should not be any single instance of a faculty taking two classes simultaneously
- A class group must not have more than one lectures at the same time

BACKGROUND

Let's learn some biology first Our body is made up of trillions of **cells**. Each cell has a core structure (**nucleus**) that contains your **chromosomes**.

Each **chromosome** is made up of tightly coiled strands of deoxyribonucleic acid (**DNA**). **Genes** are segments of DNA that determine **specific traits**, such as eye or hair color. You have more than 20,000 genes

A gene **mutation** is an alteration in your DNA. It can be inherited or acquired during your lifetime, as cells age or are exposed to certain chemicals. Some changes in your genes result in
genetic disorders

Natural Selection: Darwin's theory of volution: only the organisms best adapted to their environment tend to survive and transmit their genetic characteristics in increasing numbers to succeeding generations while those less adapted tend to be eliminated.

GA is inspired from Nature

- A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators

Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions.

- Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.
- The evolution usually starts from a population of randomly generated individuals and happens in generations.
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population.
- The new population is then used in the next iteration of the algorithm.
- Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.
- If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached. The Genetic Algorithm - a brief overview Before we can use a genetic algorithm to solve a

problem, a way must be found of *encoding* any potential solution to the problem. This could be as a string of real numbers or, as is more typically the case, a binary bit string. We will refer to this bit string from now on as the chromosome. A typical chromosome may look like this:

10010101110101001010011101101110111111110
1

At the beginning of a run of a genetic algorithm a large population of random chromosomes is created. Each one, when decoded will represent a different solution to the problem at hand. Let's say there are *N* chromosomes in the initial population. Then, the following steps are repeated until a solution is found

- Test each chromosome to see how good it is at solving the problem at hand and assign a **Fitness Score** accordingly.

The fitness score is a measure of how good that chromosome is at solving the problem to hand.

- Select two members from the current population. The chance of being selected is proportional to the chromosomes fitness. **Roulette Wheel**

Selection is a commonly used method.

- Dependent on the **Crossover rate** crossover the bits from each chosen chromosome at a randomly chosen point.

- Step through the chosen chromosomes bits and flip dependent on the **Mutation rate**.

- Repeat step 2, 3, 4 until a new population of *N* members has been created.

- Keep repeating until required fitness is achieved.

DESIGN AND IMPLEMENTATION

Objects of Time Table Scheduler

- **Students Group** The StudentGroup class has the ID, name of the student group, number of subjects, array of subject names and hours of study required for each subject per week. It also contains the id of teachers who will teach those subjects.

- **Teacher** It is a class to hold the faculty information. It has an id, name of faculty, subject that he/she teaches and an in assigned which represents the no. of batches assigned to the teacher.

- **Slot** A slot here is the most basic unit of Genetic algorithm. It represents a single characteristic of a Gene.

- **Time Table** This class' object holds an array of Slot. This is basically a class to generate new slots initially for each Student group.

- **Gene** It is the main constituent of a Chromosome and is made up of a sequence of slot numbers. It represents a Time table of a single class group

- **Chromosome** A chromosome here is a collection or an array of Genes. It is the main class of algorithm and it undergoes crossover and mutation to furnish fitter individuals.

- **Utility** It is basically for testing purpose only.
It contains some mehods like print Slots() which help to keep track of algorithm through console.

Input data This is a class mainly to fetch the input from user either through text file or through form and provide it to the working classes of the algorithm.

- **SchedulerMain** This is the main class of the algorithm which invokes other classes and calls methods for crossover, mutation , selection etc.

ALGORITHM

- First of all an initial generation of chromosomes is created randomly and their fitness value is analysed.

- New Generations are created after this. For each generation, it performs following basic operations:

a. First of all preserve few fittest chromosomes from the previous generation as it is. This is called Elitism and is necessary to preserve desired characteristics in the coming generations .
b. Randomly select a pair of chromosomes from the previous generation. Roulette wheel selection method has been used here in this project.
c. Perform crossover depending on the crossover rate which is pretty high usually. Here single point crossover has been used.
d. Perform mutation on the more fit chromosome so obtained depending on the mutation rate which is kept pretty small usually.

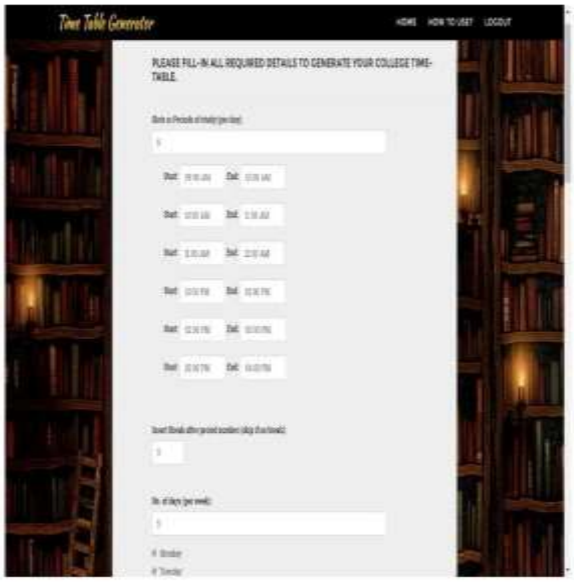
- Now analyze the fitness of the new generation of chromosomes and order them according to fitness values.

- Repeat creating new generations unless chromosomes of desired fitness value i.e. fitness=1, are obtained.

TESTING

- For the ease of testing and tracking, a lot of information is printed on the console itself. It involves input information, slots generated, few chromosomes from each generation of chromosome, fitness of these chromosomes,

- Also a sample input has been provided below the form for testing purpose during development period. It helps the developer to test the project without having to fill the complete form.



PLEASE FILL-IN ALL REQUIRED DETAILS TO GENERATE YOUR COLLEGE TIME-TABLE.

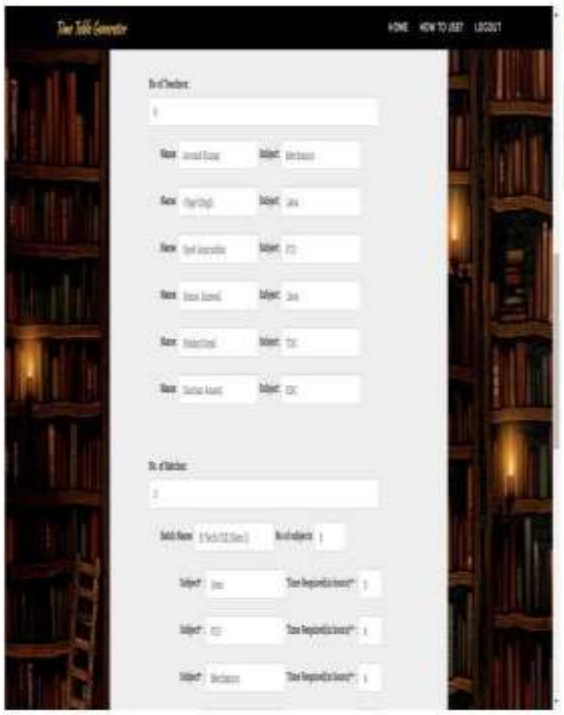
Slots or Periods of study (per day):
6

Start: [illegible] End: [illegible]
Start: [illegible] End: [illegible]
Start: [illegible] End: [illegible]
Start: [illegible] End: [illegible]
Start: [illegible] End: [illegible]
Start: [illegible] End: [illegible]

Insert Break after period number (skip if no break):
3

No. of days (per week):
5

Monday
Tuesday



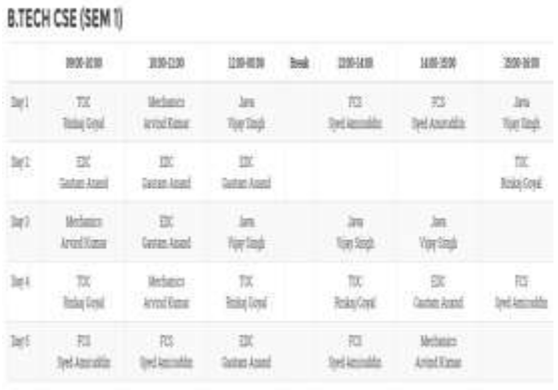
No. of Teachers:
6

Name:	Arvind Kumar	Subject:	Mechanics
Name:	Vijay Singh	Subject:	Java
Name:	Syed Aminuddin	Subject:	FCS
Name:	Sonoo Jaiswal	Subject:	Java
Name:	Pankaj Goyal	Subject:	TOC
Name:	Gautam Anand	Subject:	EDC

No. of Batches:
3

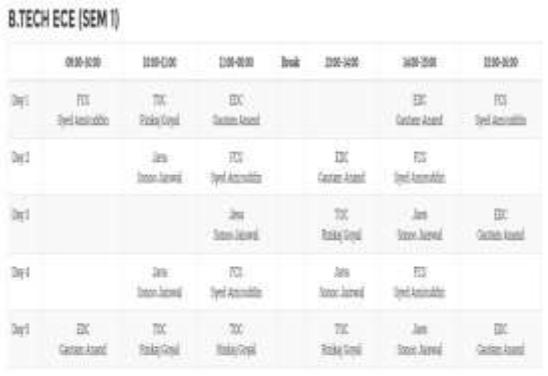
Batch Name: B.Tech CSE (Sem 1) No of subjects: [illegible]

Subject*:	Java	Time Required(in hours)*:	[illegible]
Subject*:	FCS	Time Required(in hours)*:	[illegible]
Subject*:	Mechanics	Time Required(in hours)*:	[illegible]



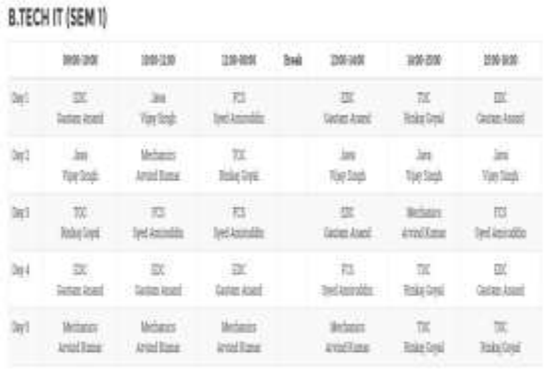
B.TECH CSE (SEM 1)

	09:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Day 1	TOC Pankaj Goyal	Mechanics Arvind Kumar	Java Vijay Singh		FCS Syed Aminuddin	FCS Syed Aminuddin	Java Vijay Singh
Day 2	EDC Gautam Anand	EDC Gautam Anand	EDC Gautam Anand				TOC Pankaj Goyal
Day 3	Mechanics Arvind Kumar	EDC Gautam Anand	Java Vijay Singh		Java Vijay Singh	Java Vijay Singh	
Day 4	TOC Pankaj Goyal	Mechanics Arvind Kumar	TOC Pankaj Goyal		TOC Pankaj Goyal	EDC Gautam Anand	FCS Syed Aminuddin
Day 5	FCS Syed Aminuddin	FCS Syed Aminuddin	EDC Gautam Anand		FCS Syed Aminuddin	Mechanics Arvind Kumar	



B.TECH ECE (SEM 1)

	09:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Day 1	FCS Syed Aminuddin	TOC Pankaj Goyal	EDC Gautam Anand			EDC Gautam Anand	FCS Syed Aminuddin
Day 2		Java Sonoo Jaiswal	FCS Syed Aminuddin		EDC Gautam Anand	FCS Syed Aminuddin	
Day 3			Java Sonoo Jaiswal		TOC Pankaj Goyal	Java Sonoo Jaiswal	EDC Gautam Anand
Day 4		Java Sonoo Jaiswal	FCS Syed Aminuddin		Java Sonoo Jaiswal	FCS Syed Aminuddin	
Day 5	EDC Gautam Anand	TOC Pankaj Goyal	TOC Pankaj Goyal		TOC Pankaj Goyal	Java Sonoo Jaiswal	EDC Gautam Anand



B.TECH IT (SEM 1)

	09:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Day 1	EDC Gautam Anand	Java Vijay Singh	FCS Syed Aminuddin		EDC Gautam Anand	TOC Pankaj Goyal	EDC Gautam Anand
Day 2	Java Vijay Singh	Mechanics Arvind Kumar	TOC Pankaj Goyal		Java Vijay Singh	Java Vijay Singh	Java Vijay Singh
Day 3	TOC Pankaj Goyal	FCS Syed Aminuddin	FCS Syed Aminuddin		EDC Gautam Anand	Mechanics Arvind Kumar	FCS Syed Aminuddin
Day 4	EDC Gautam Anand	EDC Gautam Anand	EDC Gautam Anand		FCS Syed Aminuddin	TOC Pankaj Goyal	EDC Gautam Anand
Day 5	Mechanics Arvind Kumar	Mechanics Arvind Kumar	Mechanics Arvind Kumar		Mechanics Arvind Kumar	TOC Pankaj Goyal	TOC Pankaj Goyal

CONCLUSION EVALUATION AND FURTHER WORK

CONCLUSION

The process of Time Table generation has been fully automated with this software. This web app can now cater to multiple colleges, universities and schools which can rely on it for their Time Table scheduling which earlier had to be done by hand.

EVALUATION

Using Genetics Algorithm, a number of trade-off solutions, in terms of multiple objectives of the problem, could be obtained very easily.
Moreover, each of the obtained solutions has been found much better than a manually prepared solution which is in use.

FURTHER WORK

• Though this web-app serves as a basic time table generator, there is a lot more which could be done to make this project even better in terms of consideration of soft constraints like professor giving preference to particular class.
Automated Time Table Scheduler Using Genetic Algorithm Pranav Khurana

• The up-gradations I look up to currently will be Classroom size considerations, lab facility consideration and multiple subject selection for faculty. I will try to bring the following up-gradations very soon.

• More features such as schedule print for individual faculty etc. would also be involved to make this more useful as a final product

References/Bibliography

BOOKS

- Artificial Intelligence by Stuart J. Russell and Peter Norvig
- Genetic Algorithms by David E. Goldberg

REFERENCES

- http://www.javatpoint.com
- http://www.ai-junkie.com/ga/intro/gat3.html

http://www.obitko.com/tutorials/genetic algorithms/encoding.php https://en.wikipedia.org/wiki/Genetic_algorithm
https://www.researchgate.net

- Automatic Timetable Generation using Genetic Algorithm-International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 2, February 2015