

Project report on

AUTOMATED TIME-TABLE SCHEDULER (USING GENETIC ALGORITHMS)



INDEX

- Introduction
- Background
 - Biology lesson
 - Genetic Algorithm(GA) inspired from Nature
 - Brief overview of GA
 - Outline of GA
- Design and Implementation
 - Objects of Scheduler
 - Terminologies
 - Algorithm
- Testing
- Conclusion , Evaluation and Further work
- Brief Technical Description
- References

INTRODUCTION

Time Table Scheduling Using Genetic Algorithm

Time Table Scheduling is an **NP-hard** problem and hence polynomial time verifiable using genetic algorithms. It is 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 error-prone 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in scheduling, the output is still valid, but hard constraints are those which if we violate them; the timetable is no longer valid.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) is given to every Batch

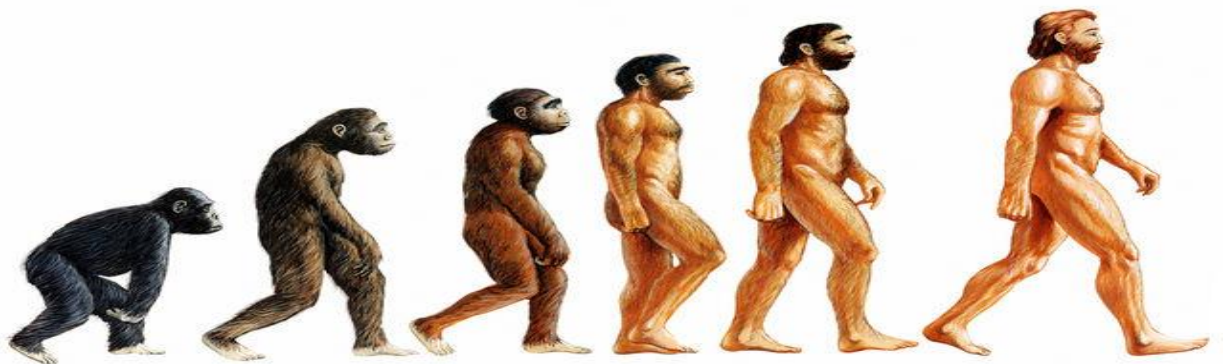
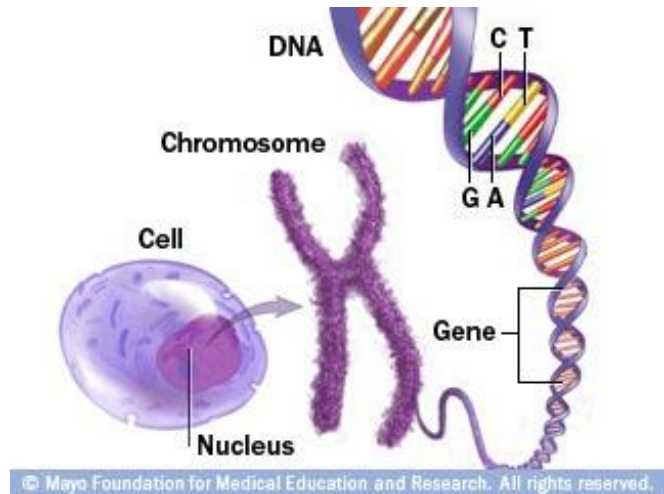
Hard-constraints (rigid):

- 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 evolution: 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:

10010101110101001010011101101110111111101

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 methods like printSlots() which help to keep track of algorithm through console.

- **Inputdata**

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.

Terminologies involved

- **Permutation Encoding**

In **permutation encoding**, every chromosome is a string of numbers, which represents number in a **sequence**. Other encoding techniques are Binary encoding Value encoding tree Encoding.

- **Elitism**

A practical variant of the general process of constructing a new population is to allow the best organism(s) from the current generation to carry over to the next, unaltered. This strategy is known as *elitist selection* and guarantees that the solution quality obtained by the GA will not decrease from one generation to the next.

- **Roulette Wheel Selection**

It is a selection procedure in which the possibility of selection of a chromosome is directly proportional to its fitness.

- **Single Point Crossover**

It is that type of crossover between two chromosomes in which the chromosomes are broken at a single point and then crossed. When single point crossover happens in this project, it is made sure that chromosome is not so cut that it intersects the timetable of any student group.

- **Swap Mutation**

It is the type of mutation technique in which the chromosomes are so mutated that two portions of the chromosome get exchanged resulting in a new chromosome.

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, maximum fitness in a generation and final selected chromosome.

```

Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk1.8.0_71\bin\javaw.exe (Nov 28, 2016, 10:38:03 PM)
45 51 36 58 63 66 56 38 62 67 40 39 53 50 37 35 47 65 60 69 46 41 64 55 57 61 54 49 44 42 43 48 52 59 68
72 84 94 101 93 82 100 95 86 85 98 78 89 102 104 83 73 77 81 70 71 96 79 90 92 74 75 99 87 80 76 88 97 103 91
125 130 121 116 128 122 127 119 137 112 139 118 106 107 110 111 120 123 117 135 115 134 133 129 113 126 131 136 138 109 132 124 108 105 114
158 145 173 147 166 140 170 148 172 165 150 164 167 174 171 160 159 169 142 168 161 152 144 141 163 162 157 156 143 155 153 146 154 151 149

Chromosome no.1: 0.9857142857142858
0 3 21 28 6 30 32 19 29 5 24 9 8 23 7 26 11 15 17 13 22 33 27 12 18 4 14 1 10 31 25 2 34 20 16
65 55 37 53 64 52 66 39 59 36 58 57 43 60 35 47 40 68 69 51 42 41 61 49 38 62 56 63 46 54 44 67 50 45 48
72 84 94 101 93 82 100 95 86 85 98 78 89 102 104 83 73 77 81 70 71 96 79 90 92 74 75 99 87 80 76 88 97 103 91
125 130 121 116 128 122 127 119 137 112 139 118 106 107 110 111 120 123 117 135 115 134 133 129 113 126 131 136 138 109 132 124 108 105 114
158 145 173 147 166 140 170 148 172 165 150 164 167 174 171 160 159 169 142 168 161 152 144 141 163 162 157 156 143 155 153 146 154 151 149

Chromosome no.2: 0.9785714285714285
0 3 21 28 6 30 32 19 29 5 24 9 8 23 7 26 11 15 17 13 22 33 27 12 18 4 14 1 10 31 25 2 34 20 16
58 46 35 55 48 52 65 68 63 37 45 41 66 61 62 39 54 51 67 43 42 49 36 53 64 44 69 40 59 60 56 57 50 47 38
72 84 94 101 93 82 100 95 86 85 98 78 89 102 104 83 73 77 81 70 71 96 79 90 92 74 75 99 87 80 76 88 97 103 91
125 130 121 116 128 122 127 119 137 112 139 118 106 107 110 111 120 123 117 135 115 134 133 129 113 126 131 136 138 109 132 124 108 105 114
158 145 173 147 166 140 170 148 172 165 150 164 167 174 171 160 159 169 142 168 161 152 144 141 163 162 157 156 143 155 153 146 154 151 149

Chromosome no.3: 0.9785714285714285
0 3 21 28 6 30 32 19 29 5 24 9 8 23 7 26 11 15 17 13 22 33 27 12 18 4 14 1 10 31 25 2 34 20 16
67 55 57 65 56 46 52 69 35 53 49 68 42 50 63 47 39 41 51 54 43 64 37 66 45 62 38 58 60 61 40 44 48 59 36
72 84 94 101 93 82 100 95 86 85 98 78 89 102 104 83 73 77 81 70 71 96 79 90 92 74 75 99 87 80 76 88 97 103 91
125 130 121 116 128 122 127 119 137 112 139 118 106 107 110 111 120 123 117 135 115 134 133 129 113 126 131 136 138 109 132 124 108 105 114
172 143 157 155 146 158 148 170 156 144 150 166 167 164 142 147 174 163 140 171 162 151 153 161 154 141 160 168 149 173 159 145 165 152 169

Chromosome no. 11 :0.9714285714285714
Chromosome no. 21 :0.9714285714285714

Most fit chromosome from this generation has fitness = 0.9857142857142858

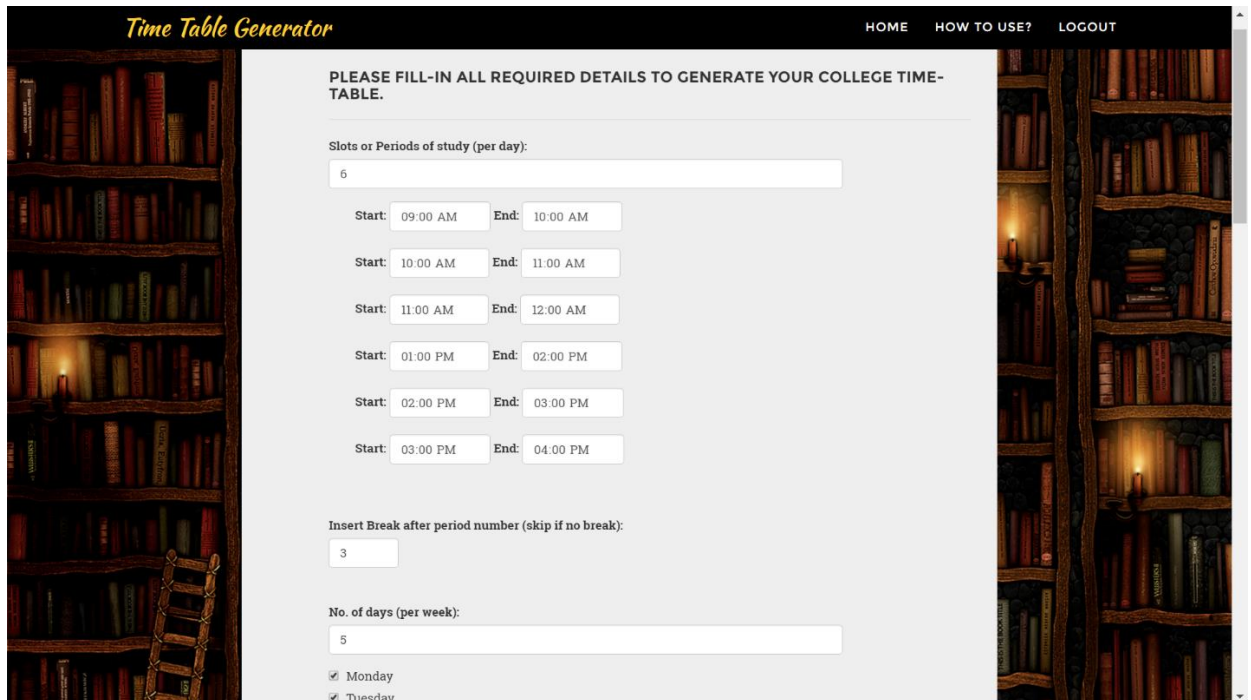
***** Generation7 *****

Fetching details from this generation...
<

```

- 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.

- When tested with new **Input** through the form for generation of time-table



Time Table Generator HOME HOW TO USE? LOGOUT

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

Slots or Periods of study (per day):

6

Start: 09:00 AM End: 10:00 AM

Start: 10:00 AM End: 11:00 AM

Start: 11:00 AM End: 12:00 AM

Start: 01:00 PM End: 02:00 PM

Start: 02:00 PM End: 03:00 PM

Start: 03:00 PM End: 04:00 PM

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

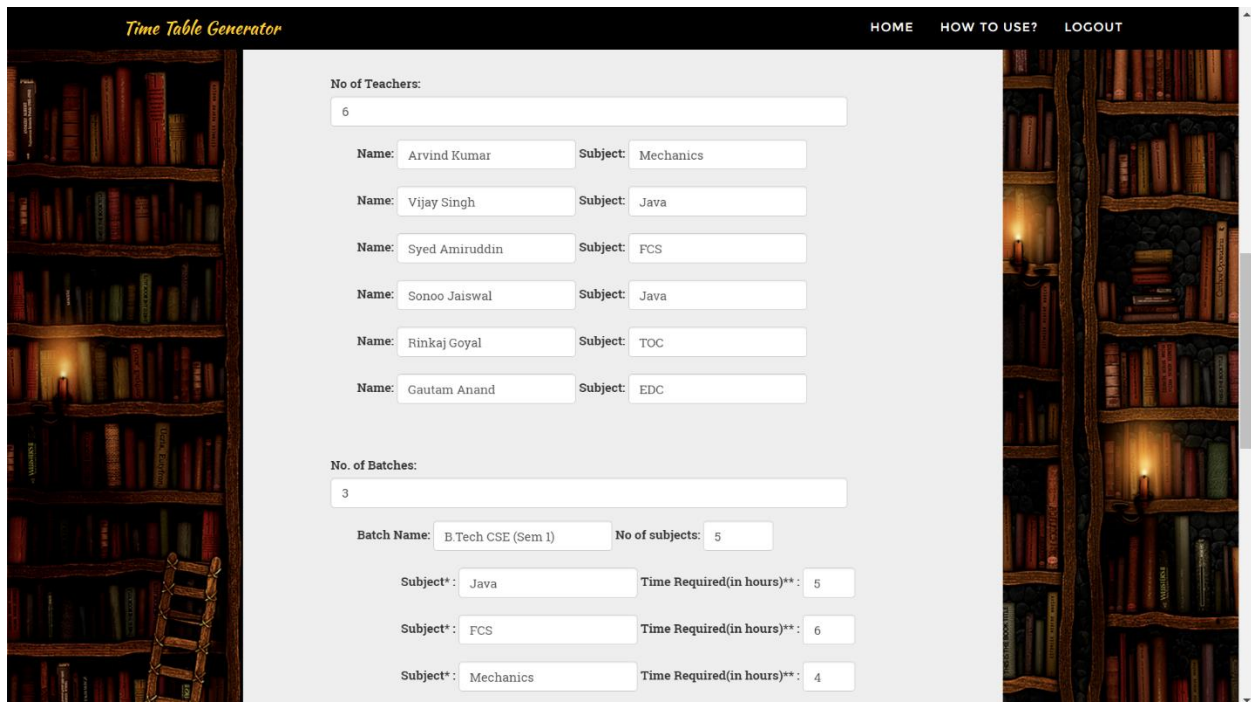
3

No. of days (per week):

5

☒ Monday

☒ Tuesday



Time Table Generator HOME HOW TO USE? LOGOUT

No of Teachers:

6

Name: Arvind Kumar Subject: Mechanics

Name: Vijay Singh Subject: Java

Name: Syed Amiruddin Subject: FCS

Name: Sonoo Jaiswal Subject: Java

Name: Rinkaj Goyal Subject: TOC

Name: Gautam Anand Subject: EDC

No. of Batches:

3

Batch Name: B.Tech CSE (Sem I) No of subjects: 5

Subject*: Java Time Required(in hours)**: 5

Subject*: FCS Time Required(in hours)**: 6

Subject*: Mechanics Time Required(in hours)**: 4

The final **Output** timetable was:

B.TECH CSE (SEM 1)

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

B.TECH ECE (SEM 1)

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

B.TECH IT (SEM 1)

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

- 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.

Technical Description

Technologies used in making the project :

- **Backend and Algorithm**

- Java 8
- Struts-2 framework
- Java Server Pages
- Servlets

- **Database**

- MySql database

- **Frontend Design**

- HTML 5
- Cascading style sheets (CSS)
- Javascript
- Bootstrap
- Ajax

The web-app has been locally hosted during development using Apache Tomcat-7.

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