**Semi-Automated Timetable Generator**

* **Customized for JIIT**

Contents

[1. Introduction 3](#_Toc402710733)

[1.1 What is a Semi-Automated Timetable Generator? 3](#_Toc402710734)

[1.2 What is a Good Timetable? 3](#_Toc402710735)

[1.2.1 Hard Constraints 3](#_Toc402710736)

[1.2.2 Soft Constraints 5](#_Toc402710737)

[1.3 Our Motivation for the Project 5](#_Toc402710738)

[2. Literature Survey 6](#_Toc402710739)

[2.1. List of Surveys Done for Formulation of Problem Statement 6](#_Toc402710740)

[2.2. Summary of Relevant Papers 6](#_Toc402710741)

[Title 6](#_Toc402710742)

[Authors 6](#_Toc402710743)

[Year of Publication 6](#_Toc402710744)

[Publishing Details 6](#_Toc402710745)

[Summary 6](#_Toc402710746)

[3. Results of Literature Survey 7](#_Toc402710747)

[3.1. Integrated Summary of Section 2 7](#_Toc402710748)

[3.2. Current Open Problems 7](#_Toc402710749)

[3.3. Problem Statement 7](#_Toc402710750)

[3.4. Overview of Proposed Solution Approach and Novelty/Benefits 7](#_Toc402710751)

[4. Analysis, Design and Modeling 8](#_Toc402710752)

[4.1. Overall Description of the Project 8](#_Toc402710753)

[4.2. Functional and Non Functional Requirements 9](#_Toc402710754)

[4.2.1. Functional Requirements 9](#_Toc402710755)

[4.2.2. Non-Functional 10](#_Toc402710756)

[4.3. Overall Architecture With Component Description and Dependency Details 10](#_Toc402710757)

[4.3.1. Hardware 10](#_Toc402710758)

[4.3.2. Platform 10](#_Toc402710759)

[4.3.3. Browser 11](#_Toc402710760)

[4.3.4. Technology Used 11](#_Toc402710761)

[User Interaction (Data Loading) 12](#_Toc402710762)

[ALGORITHMS 13](#_Toc402710763)

# Introduction

# What is a Semi-Automated Timetable Generator?

The Timetable Generator is one of the most popular constraint satisfaction problems in the world of Artificial Intelligence. Though ideally it can be used to solve time allocation problems through heuristic search algorithms, it is in reality a NP-hard problem. This is where the actual challenge lies.

From this, we have derived the name of Semi-Automated, signifying the possibility of overloading the system with too many constraints, leading to the results of only a partial solution. However, we will strive to achieve a highly-optimized and near-about complete solution using the concepts of Genetic Algorithms.

* 1. What is a Good Timetable?  
       
     Looking into the depth of the college’s constraints, a well-rounded solution can only come once we have identified the two kinds of constraints – hard and soft.
     1. Hard ConstraintsGiven below are the identified constraints we will be considering in this project -

1. Teacher can only be at one place at a time.
2. A batch can only attend one lecture at a time.
3. A classroom can only hold one subject at a time.
4. Backlog student can only be at one class at a time.
5. Working hours per week are 44 hours, from Monday to Friday (9-5), Saturday (9-1).
6. Teacher will not teach more than 16 hours a week, no less than 13 hours.
7. Teacher will not teach more than 2 subjects.
8. Teacher must take atleast 2 lab and tut optional.
9. Lunch break 12-2 PM every day. Atleast one hour break for everyone.
10. Labs accommodate only certain subjects.
11. 4 batches – bigger rooms.
12. 3 batches – smaller rooms.
13. Single batch – lab / tut.
14. Double batch tut – JBS.
15. Entire subject should not be taught in one day.
    * 1. Soft ConstraintsGiven below are the identified soft constraints which we will be considering in this project once the hard constraints are satisfied in a solution -
16. Lecture of subject only once a day.
17. Lecture must be preceded by 1 hr gap.
18. Subject preferences by teacher
19. Working hour preferences by teacher
20. Don’t teach for more than 3 continuous hours
21. Classroom preferences
    1. Our Motivation for the Project  
       Our motivation for this project comes from the understanding of how tediously manual timetable generation is done at our college, and the wish to build a solution which can put all hassles to an end. Currently the timetables at college are generated over a period of days, and sometimes over a week. With an automated solution, the timetable will now require only a few minutes to generate a solution, and will need minimum.

# Literature Survey

## List of Surveys Done for Formulation of Problem Statement

## Summary of Relevant Papers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Authors | Year of Publication | Publishing Details | Summary |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Results of Literature Survey

## Integrated Summary of Section 2

## Current Open Problems

The problems that are currently affecting the project are as follows -

1. **DATA LOADING**
2. Data loading must be complete, accurate and refined.
3. Data loading must not have noise.
4. Data must not be inconsistent.
5. **CONSTRAINTS**
6. All the constraints must be fully identified, both hard and soft.
7. Correct weightage must be given to the constraints.

## Problem Statement

The project analyzes the nature of the college’s academic timetable as a NP-hard problem, which is difficult to solve by the conventional approaches. With a high number of constraints to be applied over the benchmark set of data, the problem can achieve its optimal solution using the genetic algorithm approach.

## Overview of Proposed Solution Approach and Novelty/Benefits

There are several algorithms that can be used for timetable scheduling, that have the following set of arguments while decision-making-

1. **Backtracking Algorithm**

The important feature of this approach is solution space generated in search while implementation. However the drawback is the high time complexity. It is preferable to use with the combination of other algorithms.

1. **Graph Coloring**
2. **Genetic Algorithm**  
   It is an iterative, adaptive heuristic approach which holds for global optimization. It is a quick, simple, fault-tolerant algorithm, where the search doesn’t act directly on the structure rather on the individual parameters. However, the problem arises when there is early convergence over a set of two similar parents, where the results obtained are same and can cause termination before achieving the actual optimized solution.
3. **Pooling Algorithm**
4. **Our Approach**
   1. **Task Division Among Group Members**Research work and implementation strategies were a collective group effort, wherein each member studied his/her own set of research papers and proposed strategies. Finalizing the implementation algorithm was done collectively. Thereafter, data loading retrieved from the user and college was implemented by Anjali and Vasundhra using web interface. Data retrieval from the system was implemented by Utkarsh using C++. The report was prepared by all the group members.

# Analysis, Design and Modeling

## Overall Description of the Project

The project is being implemented in the form of a web application, involving the following features –

1. **Data Entry by Admin**  
   The admin has the right to input the academic details of the college, including Subjects Offered, Classrooms (Lecture, Tutorials and Labs) available and Year-Wise Batches.
2. **Data Entry by Teacher**  
   Each teacher will be offered a separate login and profile where they can enter their preferences for subjects they’d like to teach, time preferences and classroom preferences.
3. **Data Allocation**  
   At the backend, once the data has been completely retrieved, the system will automatically generate an algorithm to divide the batches, allocate them to each teacher depending on available number of teachers as well as subjects required.
4. **Data Distribution**  
   Using the Genetic Algorithm, we will be generating the optimal timetable solution. This will be also be done at the backend, and final results when ready, will be reflected back onto a separate page accessible by teachers.

## Functional and Non Functional Requirements

### Functional Requirements

Requirement ID : R1.01  
Title : INSERT COLLEGE DATA  
Description : The admin will fill the subjects, courses available, classrooms and the batches details for the existing academic year.  
Priority : 1

Requirement ID : R1.02  
Title : VIEW COLLEGE DATA  
Description : The admin can view the subjects, courses available, classrooms and the batches details for the existing academic year. He/She can also view the teachers’ details.  
Priority : 1

Requirement ID : R1.03  
Title : EDIT COLLEGE DATA  
Description : The admin will edit the subjects, courses available, classrooms and the batches details for the existing academic year.  
Priority : 1

Requirement ID : R2.01  
Title : INSERT PERSONAL DATA  
Description : The teacher will fill the personal information, subject preferences and timeslot availability details for the existing academic year.  
Priority : 1

Requirement ID : R2.02  
Title : VIEW PERSONAL DATA  
Description : The teacher can view the personal information, subject preferences and timeslot availability details for the existing academic year.  
Priority : 2

Requirement ID : R2.03  
Title : EDIT PERSONAL DATA  
Description : The teacher can edit the personal information, subject preferences and timeslot availability details for the existing academic year.  
Priority : 2

Requirement ID : R3.01  
Title : EDIT TIMETABLE  
Description : The timetable committee has the right to view and edit the semi-generated timetable.  
Priority : 1

Requirement ID : R4.01  
Title : VIEW TIMETABLE  
Description : Any user has the right to view the generated timetable according to various filters such as personal TT, subject-wise, year-wise, room-wise, batch-wise, etc.  
Priority : 2

### Non-Functional

#### Accessibility

The application should be accessible from all locations, and should be supported with multiple filters on the basis of batch, personal timetable, year-wise TT or subjects.

#### Disaster Recovery

Data should have backup and quick recovery.

#### Efficiency

The application should be responsive to user requests and should generate results in human sparing time.

#### Extensibility

The application should have the scope for further functions with parameters such as additional time space and constraint adjustment.

#### Maintainability

Being a web application, maintenance of the system should be minimum and things such as data inputting should require bare minimum human surveillance after initial inputs.

#### Reliability

The timetable should be able to make good of the maximum number of constraints to be reliable.

#### Security

The user data should be protected from manipulation or intrusion.

#### Usability

The application should be easy-to-use and navigate; it should be responsive and quick.

## Overall Architecture With Component Description & Dependencies

### Hardware

The project is hardware independent. It does not require any special hardware specifications or installations.

### Platform

The project is platform independent; it will run on all operating systems from Windows, Mac Linux to Ubuntu.

### Browser

The project should function on all browsers including IE, Google Chrome, Mozilla Firefox and Safari on all grounds from CSS to JS, and PHP.

### Technology Used

The first technical choice that we opted for was algorithm implementation in C++. In order to generate a user-friendly application, we have used front-end data loading and user interaction to build a web application using the technologies HTML, JavaScript, CSS and PHP.

## Proposed Algorithm

## Risk Analysis and Mitigation Plan

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Risk Area** | **(P)** | **(I)** | **RE** | **Mitigating? (Y/N)** | **M. Plan** | **Con. Plan** |
| 1. | Data Redundancy | Data Loading | 3 | 5 | 15 | Yes | Data validation | N/A |
| 2. | Overly constrained | TT Generation | 5 | 5 | 25 | No | N/A | Manual allocation |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

### Test Plan

### Implementation

Appendix

1. Project Plan
2. Details of New Tools
3. References

# User Interaction (Data Loading)

**Inputs**

1. Teacher
   1. Personal Info (Name, Email)
   2. Subjects that can be taught
   3. Timing Preferences
2. Batch
   1. Batch info (Name, Year)
   2. Subjects being taken
3. Subject
   1. Subject Info (Name, Code, Department, Duration, Type)
4. Classrooms
   1. Classroom info (Name, Capacity, Type, Subject Constraint)
5. Backlog Students
   1. Student info (Enrolment Number)
   2. Batch info (Batch, Year)
   3. Subject

# ALGORITHMS

**Ranking**

1. Backlog student
2. First year > Second Year > Third Year > Fourth > Fifth Year
3. Lectures > Labs > Tutorials

**Algorithms**

1. **Basic Workflow Breakdown**  
   INPUT college data  
   INPUT teacher data  
   ASSIGN subject TO teacher  
   ASSIGN batches TO teacher  
   **ASSIGN tentative class TO teacher+batch(s)**  
   ASSIGN (time AND room) TO (teacher+batches)
2. **Assigning Subject to Teacher**  
   Teachers Required = Batches Studying Subject / (4 or 3)  
   If (teacher can teach subject AND Subjects being taught by teacher < 3 )

Teacher -> subject

1. **Assigning Lecture to Teacher**

CREATE lecture-groups (4 or 3 batches each)

If (group studies subject)

Teacher -> lecture

1. **Assigning Lab to Teacher**

If (teaching hours < 30 AND teacher -> lab == 0)

Teacher -> Lab

1. **Assigning Tut to Teacher**

If (teacher can teach subject AND teaching hours < 30)

Teacher -> tut

1. **Assigning Classroom AND Time**

CREATE list of classes that are suitable for (teacher + subject + batch)

CREATE subject-wise backlogs

**Testing**

1. Suppose too many backlog students
2. Penalty should decrease, fitness equation should be more optimized after every execution
3. Rooms decreased for some reason
4. References

[1] “Optimal Time-Table Generation by hybridized Bacterial Foraging and Genetic Algorithms”, by Om Prakash Verma, Rohan Garg and Vikram Singh Bisht, 2012  
  
[2] “A Differential Evolution Algorithm With Dual Populations for Solving Periodic Railway Timetable Scheduling Problem” by Jing-Hui Zhong, Student Member, IEEE, Meie Shen, Jun Zhang, Senior Member, IEEE, Henry Shu-Hung Chung, Senior Member, IEEE, Yu-Hui Shi, Senior Member, IEEE, and Yun Li, Member, IEEE, 2012  
  
[3] “Exam Timetabling Using Graph Colouring Approach” by Burairah Hussin, Abd Samad Hasan Basari, Abdul Samad Shibghatullah, Siti Azirah Asmai, 2011  
  
[4] “A Curriculum Scheduling System Based on Pooling Algorithm” by FENG LU , 2011  
  
[5] “The Mathematical Model of Course Scheduling and Its Realization Based on Genetic Algorithm” by Fengmei Xie, 2011  
  
[6] “A Two Phase Integer Linear Programming Approach to Solving the School Timetable Problem” by Samir Ribi, Samim Konjicija, 2010

1. “An Evolutionary Algorithm Hyper-Heuristic for Producing Feasible Timetables for the Curriculum Based University Course Timetabling Problem”, by Rosanna Els, Nelishia Pillay, 2010
2. “The Application of a Parallel Genetic Algorithm to Timetabling of Elementary School Classes” by Nedim Srndic, Emir Pandzo, Mirza Dervisevic, Samim Konjicija, 2009
3. “USING A GENETIC ALGORITHM OPTIMIZER TOOL TO SOLVE UNIVERSITY TIMETABLE SCHEDULING PROBLEM” by Sehraneh Ghaemi, Mohammad Taghi, Vakili Ali Aghagolzadeh, 2007
4. “Towards Constraint-Informed Information Systems”, Irene Rodrigues, Nuno Mato, Salvador Abreu
5. “DESIGN OF AN EXPERT SYSTEM FOR IT COLLEGE COURSE TIMETABLING AT THE UNIVERSITY OF BAHRAIN BY USING A KNOWLEDGE BASE PROCESS MODELING” by Maqbool U. Shaikh, Yousif Al-Bastaki i, 2004