

# Automated Timetabling System for University Course

Mrunmayee V. Rane

Electronics and Telecommunication,  
 Thadomal Shahani Engineering College,  
 University of Mumbai,  
 Maharashtra, India

Vikram M. Apte

Electronics and Telecommunication,  
 Thadomal Shahani Engineering College,  
 University of Mumbai,

Vishakha N. Nerkar

Thadomal Shahani Engineering College,  
 University of Mumbai,  
 vishakhanerkartechpaper@gmail.com

Mani Roja Edinburg

Electronics and Telecommunication,  
 Thadomal Shahani Engineering College,  
 University of Mumbai  
 Maharashtra, India  
 maniroja@yahoo.com

K.Y. Rajput

Electronics and Telecommunication,  
 Thadomal Shahani Engineering College,  
 University of Mumbai  
 Maharashtra, India

**Abstract**— Generating timetable for university with many branches, different years, and multiple batches is a tiresome job. It takes a lot of time and is required for every semester that is of six months. Thus, a lot of time and manpower is taken up by this process. In some cases, the manual process of creating timetables is too cumbersome. Creating temporary timetables when a faculty is on leave is impractical. In this paper, we have created an algorithm to generate timetables which can save a lot of time and pressure on the person doing this job manually. A software-based approach is better, as a computer can process data and churn out results at a much higher rate with greater accuracy. Also, it can be coded and optimized as per requirements set by different universities and the base code will remain the same.

**Keywords**— Automated Timetabling, timetable, soft constraints, hard constraints, optimization, python, TKinter toolbox, CSV (comma separated values) file.

## I. INTRODUCTION

“Timetabling is one of the common scheduling problems, which can be described as the allocation of resources for tasks under predefined constraints so that it maximizes the possibility of allocation or minimizes the violation of constraints” [1]. Due to the non-proportionality between a rapid increase in the number of students enrolled, university courses and accessible resources, the complexity of the problem has become multifold. “This problem is described as a Non-Polynomial-hard combinatorial optimization problem having no specific solution” [2].

Currently, in most of the institutes, this time-consuming most complex problem is tackled manually by a single person or a group of related people with the only goal of producing feasible timetables. This may lead to unnecessary wastage of faculties and concerned members precious time due to lack of optimization in the generation of timetable. One cannot deny the human errors which will introduce conflicts such as overlapping of resources, double booking of time-slots, etc.

To generate conflict-free and optimized timetable, adequate constraints need to be considered. These constraints are often classified as either hard or soft in nature. Fulfilling just hard constraints will create a feasible solution but satisfying maximum soft constraint would give an optimized solution.

These constraints can be viewed from student’s, faculty’s and resource’s perspective.

- **Student’ s point of view** – “All my lectures must schedule in different time-slots to avoid clashes!” “I want adequate time gaps between lectures so that my time will not be wasted!”
- **Faculty’ s point of view** – “I cannot teach more than one lesson during any time slot!” “Teaching continuously more than 2 hours is exhausting for me!”
- **Resource’ s perspective** – Two or more lectures, tutorial or practical cannot occupy the same Classroom/ laboratory in a common slot.

The development of the algorithm to generate conflict-free, optimized timetable automatically in less time for large university courses exemplified in this paper by using data of Electronics and Telecommunication (EXTC) department of Thadomal Shahani Engineering College (TSEC), University of Mumbai.

## II. LITERATURE REVIEW

The problem of class teacher timetabling was first studied by Gotlieb in 1962. [3] Since then several algorithms have been introduced to solve the problem. The earliest solution proposed was based on sequential methods that deal with timetabling problems as a graph problem. [4] Later, several researchers applied the Evolutionary Algorithm (EA) and Genetic Algorithm (GA) approach to find a feasible and optimized solution for highly constrained systems [5]. Meta-heuristic methods [6], tabu search, cluster methods, simulated annealing, scatter search methods, fuzzy logic [7] are some other approaches [8] used by the intelligent methods [9] such as Swarm intelligence, [10] Artificial Neural Networks and hybrid approach. However, all of these methods took help of foundational algorithms which are highly sensitive, even a small change results in different group level behaviour. Also, the user interface was not involved, which made deployment harder.

Mohamed Abdelfattah and Ahmed Shawish published a paper on ‘Automated Academic Schedule Builder for University’ which “the genetic algorithm to produce an optimal timetable for each faculty within a university. The proposed application produced a nearly optimal timetable”. The fitness function designed here is generic in nature to fit in any environment, hence require some manual modification in the end to satisfy all the requirements. The more complex environment will take more time to produce results using genetic algorithm approach.” [11]

Another research paper on ‘automated college timetable generator’ was published by Adithya R Pai, Ashwitha S, Raksha Shetty, Prof. Geethalaxmi in 2018. “In this paper, researchers have first approached the problem with Genetic Algorithm and further compared it with a Genetic Artificial Immune Network (GAIN). Results showed that GAIN is able to reach the optimal feasible solution faster than that of GA”. [12] Use of genetic algorithms and GAIN makes it computationally expensive.

In IIT Kanpur, Dilip Datta, Kalyanmoy Deb and Carlos M. Fonseca came up with the solution for a very complex system of their institute. In this work, “The potentiality of Evolutionary Algorithms (EAs) has been exploited to schedule the even-semester classes of the institute. Using a multi-objective EA-based university class timetable optimizer, a number of trade-off solutions, in terms of multiple objectives, were obtained very easily”. [13] However, this approach is restricted to their institution.

### III. PROBLEM STATEMENT

The general timetable problem can be expressed as, a multidimensional [14] assignment problem, in which available time slots are allocated to lectures and assign them several resources (faculty, students and classrooms/laboratories) such that no clashes occur with optimum usage of time. The cumbersome timetabling problem is repeated twice a year in every academic institute. The complexity of the timetabling problem of EXTC department of TSEC institute depends on the following aspects which are taken into consideration in the paper:

- **Semester pattern:** The whole engineering course is divided into eight semesters (even and odd) of six months. In the first half of the academic year, students complete odd semesters, in the next half even semesters.
- **Dividing into batches:** Due to a large population of students and limitations of resources, each batch is divided into multiple batches according to number students in a class for practical and tutorial courses.
- **Multiple faculties:** Single-subject can be taught by more than one faculty.
- **Departmental level elective subjects:** Courses according to their interests from V semester onwards. These options are distinct for different departments.
- **Institutional level electives:** Students are given options to choose courses according to their interests

from VII semester onwards. These options are the same for different departments.

- **Fixing lecture in fixed timeslot:** Time slots and rooms of some lectures are fixed. No other class should be taken at that time.
- **Interdepartmental courses:** Students are allowed to take some courses which are taught by other departments.
- **Practical and tutorial:** Every subject has either 2 hours of practical or 1 hour of tutorial every week.
- **Contact Hours:** Every subject has a fixed number of contact hours to complete within a week.
- **Faculty Load:** Each faculty must have a total load of 14 – 16 hours in a week.

The timetabling problem includes different constraints, these are classified into hard constraints and soft constraints. The goal is to satisfy all hard constraints and to minimize the violation of the soft constraints.

### IV. PROPOSED SOLUTION

The ‘Automated Timetabling System for University Course’ is split into three modules, Input module, Operational module and Display module.

- **Input Module:** The fundamental data regarding branch, semester, time slots, subject, faculty, classroom/laboratories, practical and electives; required for creating timetables are entered, stored and updated in this module.
- **Operational Module:** The automated timetables for students, faculties and classrooms/laboratories are generated using the input data and proposed algorithm.
- **Display Module:** Generated student’ s, faculty’ s and respective classrooms’ / laboratories’ timetables are displayed in this module through the graphical user interface (GUI).

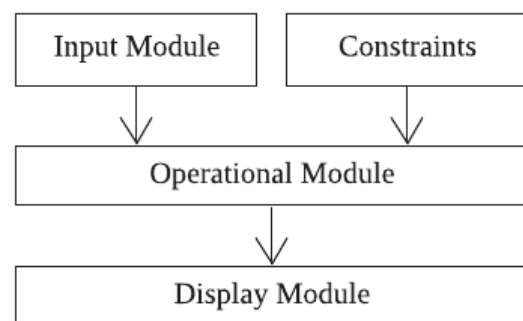


Fig. 1. System Architecture.

#### A. Graphical User Interface (GUI)

For user-friendly access to the system's input module and display module, interactive interfaces are built. These GUIs are created using Tkinter toolbox.

### B. Access to Modules

- Input Module: The authorized faculty of the department has access to the input module to preserve the integrity and authorization of the information. Access is given after verifying Login Id and Password.
- Operational Module: Access is given to only the software developing team.
- Display Module: Every student and faculty can access.

### C. Constraints

Constraints are the rules that should be followed by an operational module to create conflict-free, flexible and optimized timetables.

- 1) Hard Constraints: These are the set of rules that must be realized to construct a conflict-free timetable. The following hard constraints are taken into consideration.
  - Students must not have two lectures/practical at the same time.
  - All batches should be allotted practical if falling between two lectures.
  - There should not be a free slot other than a working break.
  - Faculty should not be given two different classes at the same time.
  - Each faculty has their minimum and maximum limit of weekly working hours.
  - Rooms and laboratories should not be double-booked in the same time-slot.
  - Weekly load of every subject should meet the university subject requirement.
  - Each lecture/practical must be assigned to its appropriate room.
  - Elective lectures in the same time slot must be allotted to different rooms.
- 2) Soft Constraints: Conditions those are optional but advocated to achieve more enhanced timetable for the faculty and students.
  - All parameters are customized by the user.
  - No two tutorials in one day so students can prepare properly.
  - Faculty should not have consecutive lectures in a day.
  - Lectures should be scheduled in one building or at one location on campus to minimize delays and avoid exhaustion for students and faculties.

- Some classes need to be held consecutively. For example, a four-hour-long practical.

### D. Input Module

The option 'Enter Details' will direct the user to an input module after validating login ID and password. The following details are taken from the user and stored into CSV files. The details are entered in the input module as shown in Fig. 2.

Fig. 2. Input Module

- 1) Branch Details: Branch name for which the timetables should be generated is selected.
- 2) Working Days: Select all the working days of the week.
- 3) Time-Slots Detail: The start and end time of working slots and break slots in a day are entered.
- 4) Subject Details: The name of each subject with a semester in which it is taught, its weekly theory, practical and tutorial load, classroom and laboratory details are provided.
- 5) Faculty Details: Faculty name, subject which he/she is going to teach, total theory, practical and tutorial load shared by him/her is given. These faculty details are entered multiple times if he/she teaches multiple subjects.
- 6) Elective Details: The name of elective subjects, the semester in which they are taught with their theory load, practical load, classroom and laboratories details are provided. The type of elective (departmental level/institutional level) is entered.
- 7) Special Case Input: Interdepartmental subjects/practical, institutional level electives, two-hour-long tutorial, etc. such as exceptional cases or any new changes to the system can be added manually to maintain the flexibility of software. The fixed time-slot with faculty and classroom/lab details are provided in such cases. The special case inputs are shown in Fig. 3.

Sr No.	Subject Name	Allotted Slots	Class room/Lab	Professor Name
SC0	Mini Project	Wednesday, 3-5 pm	705	NP
SC1	Mini Project	Friday, 3-5 pm	705	NP
SC2	Project	Monday, 3-5 pm	-	None
SC3	Project	Thursday, 3-5 pm	-	None
SC4	BDA	Thursday, 3-5 pm	609	SI
SC5	Institutional Elective	Friday, 2-5 pm	901, 502, 401	AM, PS, AJK

Fig. 3. Special Case Input

### E. Operational Module

1) **Initialization:** An empty timetable for each faculty, classroom, laboratory and students are initialized. The inputs are fetched from the input module in the required format.

2) **Special Case Allocation:** First, the special case inputs are allocated as fixed slots according to the flowchart in Fig 4.

3) **Removing Extra Slots:** To satisfy the hard constraint there shouldn't be free slot between two working slots except for breaks', delete the extra available slots. Extra slots are removed according to the flowchart in Fig. 5.

- Count the number of extra slots.

Number of extra slots = Total working slots - Total workload

- Choose random marginal slot. Marginal slot: \*Time-slot which does not have any available working slot before or after it.
- Check if it is available. If available proceed to step 4, else go to step 2.
- Remove the slot.
- Mark the adjacent slot of the removed slot as a marginal slot.
- Decrement extra slot count.
- Repeat steps 1 to 4 until the count becomes zero.

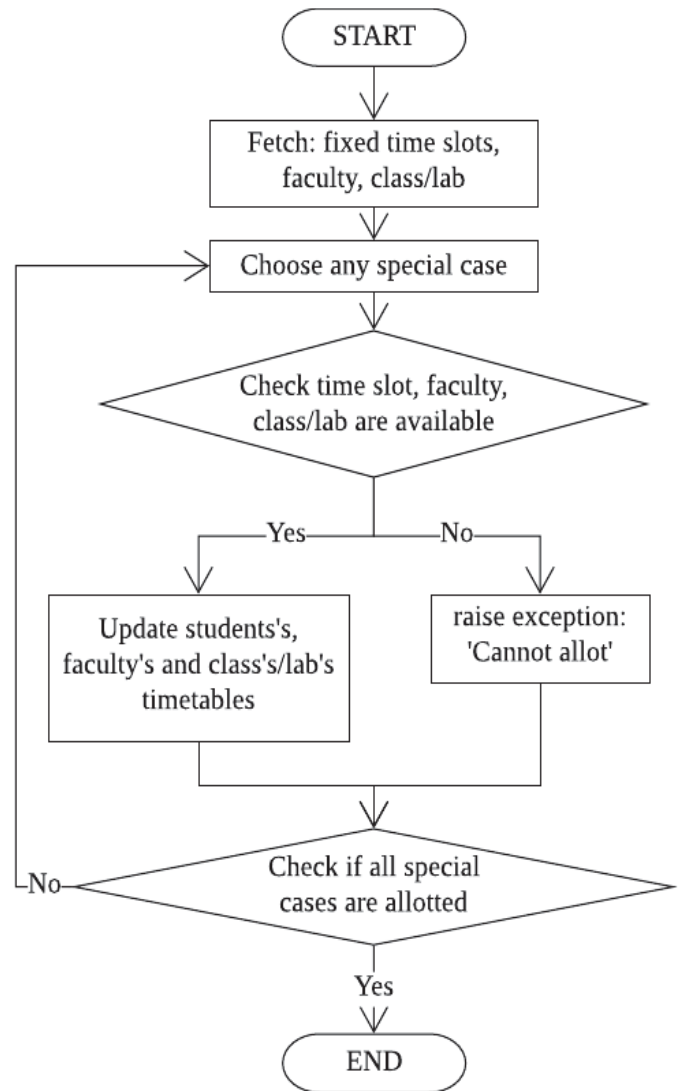


Fig. 4. Special Case Allocation Flowchart

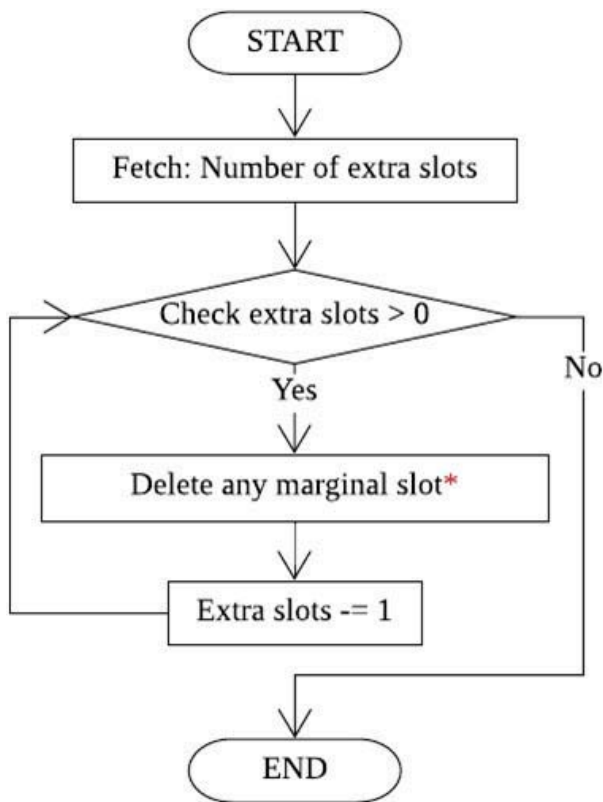


Fig. 5. Removing of the Extra Slots.

4) **Elective Allocation:** Electives are allocated according to the flowchart in Fig. 6.

1. Choose random slot from student timetable.
2. Check for slot's availability. If available go to step 3 else 1.
3. Select the same type of elective subjects from the elective list whose workload is non-zero.
4. Check for faculties and classrooms availability from faculty and classroom timetable. If available proceed to step 5, else step 1.
5. Allot the electives subject to the slot and update the respective changes into faculty and class timetable.
6. Decrease the respective subject load and faculty load by 1 hour.
7. Repeat the steps 1 to 6 until all elective theory hours become zero.
8. Finish.

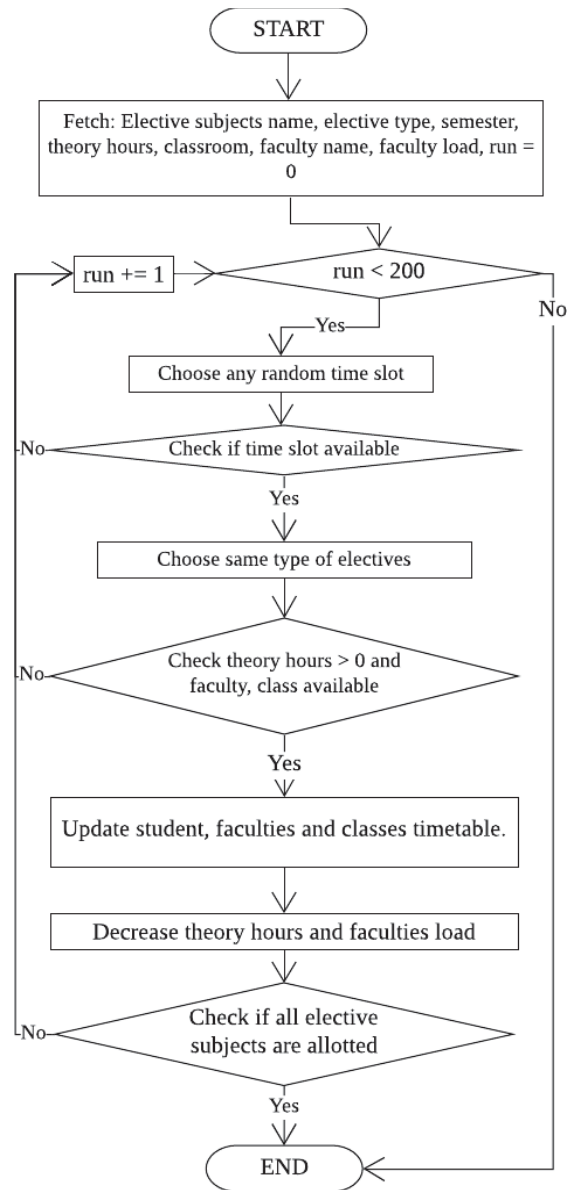


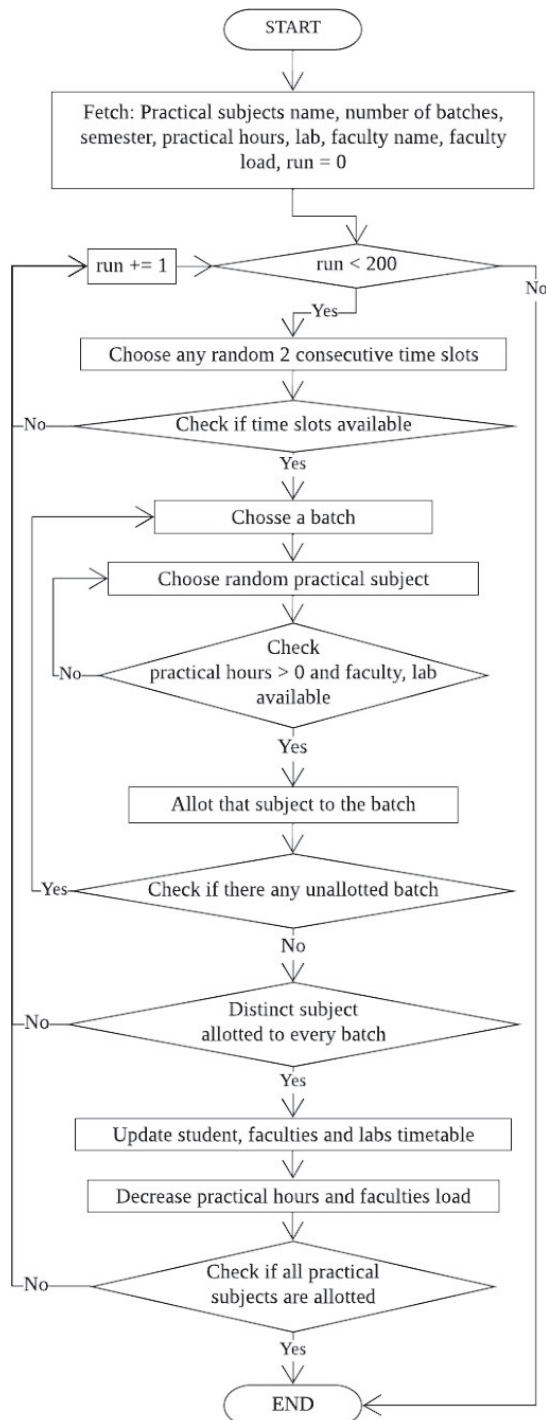
Fig. 6. Elective Allocation Flowchart

5) **Practical Allocation:** Practical slots are allocated according to the flowchart in Fig. 7.

1. Choose random two consecutive time-slots.
2. Check for slot's availability. If available go to step 3 else step 1.
3. Choose a batch.
4. Choose random practical subject whose workload is non-zero.
5. Check for faculty and lab availability from faculty and lab timetable. If available proceed to step 6, else step 4.
6. Repeat step 3-5 for every batch.



7. If all batches are allotted to distinct practical subjects, go to step 8, else step 1.
8. Allot the practical slot in student, faculty and lab timetable.
9. Decrease the respective practical and faculty workload by 2 hours.
10. Repeat the steps 1 to 9 until all the practical subjects are allotted.
11. Finish.



6) **Theory Lectures Allocation:** Theory lectures are allocated according to the following algorithm.

1. Choose random slot from student timetable.
2. Check for time-slot's availability. If available go to step 3 else go to step 1.
3. Select a random subject from theory subject list whose workload is non-zero.
4. Check for faculty and classrooms availability from faculty and classroom timetable. If available proceed to step 5, else step 1.
5. Allot the subject and update the respective changes into faculty and class timetable.
6. Decrease the respective subject's theory load and faculty load by 1 hour.
7. Repeat the steps 1 to 6 until all elective theory hours become zero.
8. Finish.

7) **Tutorial Allocation:** Tutorial slots are allocated according to the following alorithm.

1. Choose a random marginal slot.
2. Check for time-slot's availability. If available go to step 3 else go to step 1.
3. Select a random tutorial subject from whose workload is non-zero.
4. Check for faculty and classrooms availability. If available proceed to step 5, else step 1.
5. Allot the tutorial to the slot and update the respective changes into faculty and class timetable.
6. Decrease the respective subject's tutorial load and faculty load by 1 hour.
7. Repeat the steps 1 to 6 until all elective theory hours become zero.
8. Finish.

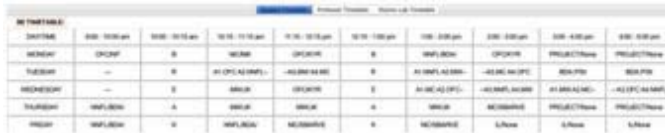
8) **Operational Module Flowchart:** Operational Module works according to the following algorithm.

1. After fetching the required details from the input module. If special case input is not empty allot them first. If not, go to step 2.
2. Remove all the extra slots.
3. If there are elective subjects in the curriculum, allot them next. If not, go to step 4.
4. If there are practical subjects in curriculum, allot them in 2-hours slots for every batch. If not, go to step 5.
5. Next, allot all the theory subjects.
6. If there are tutorials in the curriculum, allot them next.
7. Repeat the whole algorithm until all subjects are allotted.

## V. RESULTS

The proposed algorithm was programmed using python 3, uses Tkinter as a Graphic User Interface (GUI) and the algorithm was tested for timetable generation of Electronics and Telecommunication department. The GUI is user friendly and simplified; it contains a help section to guide you through the application. The algorithm after implementation leads to the generation of a timetable for a class of students (Fig 8), faculties (Fig 9), classrooms and laboratories. It also addresses all hard constraints such as different laboratories for all the batches and availability of faculties.

The soft constraints of the algorithm are effectively handled. Our algorithm was capable of finding a feasible timetable for the second year, third year, and final year with respective faculty and classroom timetables. The average iterations taken to generate all timetables were 1320. The Intel i5- 6200U processor was used; maximum process utilization was 30%. The average time was 17 sec and RMS time was 22sec, for 100 runs.



STUDENT	08:00-09:00am	09:00-10:00am	10:00-11:00am	11:00-12:00pm	12:00-01:00pm	01:00-02:00pm	02:00-03:00pm	03:00-04:00pm	04:00-05:00pm
SECTION	SECTION	SECTION	SECTION	SECTION	SECTION	SECTION	SECTION	SECTION	SECTION
TEACHER	---	---	---	---	---	---	---	---	---
LABORATORY	---	---	---	---	---	---	---	---	---
TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING
LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR

Fig. 7. Student Timetable



Prof. Name	08:00-09:00am	09:00-10:00am	10:00-11:00am	11:00-12:00pm	12:00-01:00pm	01:00-02:00pm	02:00-03:00pm	03:00-04:00pm	04:00-05:00pm
SECTION	---	---	---	---	---	---	---	---	---
TEACHER	---	---	---	---	---	---	---	---	---
LABORATORY	---	---	---	---	---	---	---	---	---
TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING	TEACHING
LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR	LABOR

Fig. 8. Faculty Timetable

## VI. CONCLUSION AND FUTURE SCOPE

This paper addresses a solution for timetabling problems of complex environments having electives, tutorials, multiple student batches, practical subjects, interdepartmental courses. The solution considers a wide range of constraints divided into soft and hard, to generate faculty, classroom/laboratory and student timetables. The implementation of the 'Automated Timetabling System' in our institute, resulted in feasible timetables, which are much more optimized and accurate than if they were created manually. The complexity of the proposed algorithm is  $n^3$ . Given the generality of the algorithm, it can be further adapted to suit as per the requirements of different institutes and universities.

Moreover, the proposed algorithm saves hours of administrator's precious time which can be utilized in other fruitful things. The program based on this algorithm has been proved effective for replacing an old-fashioned, time-consuming timetabling system with an enhanced, flexible and more reliable automated system. Also, the graphical user interfaces designed for input and display modules are very convenient to save, add or delete any input data or created timetables.

This research work can be extended to be utilized by different colleges/universities and even places where the same type of timetabling is required. The program can be hooked up with college's/university's mobile app and/or website, thus making the data more accessible and notifying the concerned students/professors, if there is a change in a timetable. Letting the students and faculties control some soft constraints of timetable, like duration of a lecture, or the starting time for the day. The most interesting future direction in the development of this algorithm lies in an automatic adaption of timetable if any lecture gets cancelled or any faculty is on long leave.

## REFERENCES

- [1] N. D. Thanh, "Solving timetabling problem using genetic and heuristics algorithms," *J. Sched.*, vol. 9, no. 5, pp. 403–432, 2006.
- [2] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multi commodity flow problems," *SIAMJ.Comput.*, vol.5,no.4,pp.691–703, 1976.
- [3] Gottlieb, C.C., The construction of class-teacher timetables. In *Proceedings of IFIP Congress*, North-Holland Pub. Co., Amsterdam, 1962, 73-77. DOI: [https://doi.org/10.1016/0305-0548\(74\)90058-6](https://doi.org/10.1016/0305-0548(74)90058-6)
- [4] T. A. Redl, "On Using Graph Coloring to Create University Timetables with Essential and Preferential Conditions," *Proceedings of the 3rd Inter. Conf. on Computational and Information Sciences*, University of Houston-Downtown, pp 162- 167, 2006.
- [5] Limkar S., Khalwadekar A., Tekale A., Mantri M., Chaudhari Y. (2015) Genetic Algorithm: Paradigm Shift over a Traditional Approach of Timetable Scheduling. In: Satapathy S., Biswal B., Udgata S., Mandal J. (eds) *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)* 2014. DOI:[https://doi.org/10.1007/978-3-319-11933-5\\_87](https://doi.org/10.1007/978-3-319-11933-5_87)
- [6] Ricardo santiago-mozos , sancho salcedo-sanz ,mario deprado-cumplido and carlos bouso-no-calz\_on , a two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish university (2005)
- [7] [Arindam Chaudhuri and Kajal De "Fuzzy Genetic Heuristic for University Course Timetable Problem" *Int. J. Advance. Soft Comput. Appl.*, Vol. 2, No. 1, March 2010 ISSN 2074-8523; Copyright © ICSRS Publication, 2010
- [8] S. Abdullah, E. K. Burke and B. McCollum, "A Hybrid Evolutionary Approach to the University Course Timetabling Problem", *Proceedings of the IEEE Congress Evolutionary Computation*, Singapore, (2007).
- [9] Anmar abuhamdah, masri ayob, graham kendall, population based local search for university course timetabling problems, *applied intelligence* (2014)
- [10] Humza turabeih, salwani abdullah, barry mcollum and pual mcmullan, fish swarm intelligent algorithm for course timetabling problem, *springer- verlag berlin Heidelberg Rskt* 2010, pp. 588-595
- [11] Mohamed Abdelfattah and Ahmed Shawish "Automated Academic Schedule Builder for University's Faculties" *Proceedings of the World Congress on Engineering 2016 Vol I WCE 2016*, June 29 - July 1, 2016, London, U.K.
- [12] Adithya R Pai, Ashwitha S, Raksha Shetty, Prof. Geethalaxmi "Automated college timetable generator" *International Journal of Scientific & Engineering Research* Volume 9, Issue 4, April-2018
- [13] Dilip Datta, Kalyanmoy Deb, Carlos M. Fonseca "Solving Class Timetabling Problem of IIT Kanpur using Multi-Objective Evolutionary Algorithm"
- [14] M. W. Carter and G. Laporte, "Recent developments in practical course timetabling," *inProc. 2nd Int. Conf. Pract. Theory Automated Timetabling*, (Lecture Notes in Computer Science). DOI:<https://doi.org/10.1007/BFb005>