Automated Activity Scheduling Using Heuristic Approach

Bhoomil Dayani[1], Raj Busa[2], Divyesh Butani[3], Dr. Nirav Bhatt[4]
Smt. Kundanben Dinsha Patel Department of Information Technology,
CSPIT, Charotar University of Science And Technology, Gujarat, India
[1]20it022@charusat.edu.in,
[2]20it011@charusat.edu.in,
[3]20it012@charusat.edu.in,
[4]niravbhatt.it@charusat.ac.in

* Corresponding Author: niravbhatt.it@charusat.ac.in

Abstract. The intricacy of classroom scheduling may result from moving and organizing classrooms based on the audience's capacity, all available facilities, lecture time, and many other factors. This paper suggests a heuristic timetable optimization method to increase lesson planning productivity. This work aims to find an optimal solution to the timetabling problem, which is one of the highly constrained N-P hard problems. The need for this sort of timetabling software arises as manually designing a timetable takes too much time and effort and if an overlap occurs among the timetable, the timetable is redesigned using hit-and-error methods which have very high time costs. So, In this work, we try to develop software that will generate a timetable automatically based on the provided information. The expected main input is about teachers, classes, and subject data along with the maximum workload of a teacher in a week to generate a valid timetable. The main constraints that this software should satisfy are that a teacher should not have a lecture in more than one class at the same time slot and a class should not have more than one lecture in a given time slot. The solution which gets from this project should have to satisfy the above-mentioned constraints.

Keywords: TimeTable scheduling, Genetic algorithm, Constraints, Optimization, heuristic.

1 Introduction

The proposal is made in "Time Table Scheduling using Genetic Artificial Immune Network." One of the crucial duties that arise in everyday life is scheduling. Numerous scheduling issues exist, including those involving employees, production, education, and others. Due to the various constraints that must be met to find a workable solution, organizing educational schedules may also be challenging. According to Jonas and Rasmus[8], the complexity of the problem is what leads to the adoption of so many different algorithms. Almost every school has unique restrictions and requirements that must be met. The phase of production

control known as scheduling assigns a priority rating to each work item and then arranges for its timely and orderly release to the plant. According to Jha (2014)[20]. Heuristic algorithms have been utilized in research with varying degrees of effectiveness[6]. Following an examination of our institute's scheduling system, we attempted to use a genetic algorithm to resolve it. In educational institutions, finding a workable lecture/tutorial schedule for a department may be a difficult issue that arises frequently[7]. For this, This work employs a special algorithm. And suggested using a timetable object in our approach to creating timetables. Even though most faculty organization tasks are now automated, creating lecture schedules is still typically done manually, which requires a lot of time and effort. In addition to being utilized extensively in schools, colleges, and other teaching settings. In these scenarios, precisely designed timetables are reused for the entire generation without any alterations, which is dull. Another example is That difficulty arises when there aren't enough employers or employees, which forces schedule adjustments or the quick filling of vacant seats. They must plan their course so that it fits the requirements of the current time frame and the facilities at their disposal. However, they must adjust their schedule to accommodate both the new course additions and the newly enrolled students in the new batches. This can lead to rescheduling the entire timetable for all of its batches to schedule it as soon as feasible before the batch courses begin. Another issue that comes up when establishing exam schedules. When many batches are scheduled to take tests on the same day, scheduling must be done carefully to account for any issues with the facilities that are available to hold these exams concurrently[21].

2 LITERATURE SURVEY

Literature surveys evaluate the data in the literature about a proposed piece of work. A literature review, which summarises all prior research on a topic and establishes the framework for ongoing research, may be an essential component of a research endeavor. It is the most crucial section of the report since it directs research most appropriately. It helps to set a goal for analysis.

The issue of time scheduling is resolved via evolutionary methods. Methods like genetic algorithms and evolutionary algorithms have been employed with varying degrees of effectiveness. In this essay, we have examined the issue of using a genetic algorithm to schedule an instructional timeline. We also used a synthetic genetic defensive network and a mimetic hybrid algorithm to tackle the issue, and we compared the outcomes with those of the genetic algorithm[1]. The results show that GAIN can reach a possible solution faster than that GA[19].

Academics frequently struggle with figuring out the study plan that is feasible in the university's major department. This study provides an evolutionary algorithm (EA) method for resolving the robust timetable problem at the institution. On to chromosomal representation issues. Utilizing timelines, heuristics, and

contextual-based reasoning may have been acquired at the appropriate computer moment[2]. To increase cohesiveness, a clever genetic alteration plan has been implemented. Using actual data from a top university, the comprehensive curriculum plan described in this paper is accepted, assessed, and discussed.

An automated timeline system employs Anuja Chowdhary's[3] efficient timing algorithm, which can manage both strong and weak impediments. so that each teacher and student can review their schedule once a specific semester is over, but without making any specific plans. By the teacher's schedule, the availability and power of visual resources, and other rules applicable to different classes, semesters, teachers, and grade levels, the Timetable Generation System develops a timetable for each class and teacher.

Anirudha Nanda[4] proposes a widespread remedy for the time issue. Most of the previously suggested heuristic programs were considered to be challenging from the viewpoint of students. However, this technique works from the perspective of the subject, i.e., the instructor's availability at a specific time. The planning strategy given in this study is adaptable, with the main goal of addressing academic and academic conflict, and teacher-related concerns, even though all potential barriers (e.g., teacher availability, etc.) are dealt with firmly.

Al-Khair[5] presented algorithmic solutions to address the scheduling issue while offering teacher availability admissions. The challenge of scheduling school time is completely resolved by this technique, which employs a heuristic methodology. It initially builds a temporary timeline using title sequences that are produced at random. The subjects are moved to the Clash data structure if a teacher is divided by more subjects than are permitted.

3 PROBLEM STATEMENT

The challenge of timetable creation can be modeled as a restricted satisfaction problem with several ambiguous factors. Modeling these problems in a way that the scheduling algorithm can understand is necessary. Planning entails setting up several sensible constraints on how many jobs can be completed at once. For instance, two courses taught by the same faculty member might not be scheduled at the same time at a tertiary school to better manage classrooms. The two disciplines that the same set of students must take also shouldn't be incompatible.

4 RELATED WORKS

The timetable planner assists students in creating their schedules based on the courses they choose to enroll in by accurately recommending the best potential

combinations for that particular semester[9]. A review of university scheduling by Burke et al.[10] focuses on meta-heuristics, multi-criteria, case-based reasoning, hyper-heuristic, and self-adaptive techniques. The goal is to improve real-world timetabling systems by using parameters, multi-criteria approaches, and other techniques. It does not rely on relevant knowledge and represents a more adaptable method for finding the best solution to more challenging challenges. New challenges are solved using the knowledge-based paradigm of case-based reasoning.

To overcome the issue and fine-tune the parameter settings, the single-stage simulated annealing (SA) design method is reliable and efficient[11]. Based on the provided set of operating parameters for timetable construction utilizing an evolutionary strategy, tabu search (TS) accelerates the solution-finding process[12]. Simulated annealing is dependent on a cooling schedule, and a long enough Tabu list is needed for Tabu search.

For the different search techniques such as Tabu Search, Simulated Annealing, Scatter search, and Genetic algorithms, Anirudha Nanda et al.[13] use a heuristic approach to try to obtain a decent approximation solution. Memetic and genetic algorithms, as well as more recent heuristics, aid in the exploration of neighborhood solutions.[14]. The runtime of the novel greedy heuristic algorithm is greatly shortened since it models the restrictions based on the objective function for all of a student's curricula and transforms heuristics using relational calculus[15]. When there are problems with conflicts between lectures and teacher-related topics, it offers an all-encompassing solution.

A sound set of solutions is produced by a heuristic strategy that doesn't break strict limitations. The cycle of Analysis, Selection, Mutation, Prioritization, and the building is followed by the evolutionary squeaky wheel optimization process, which ends when the stop condition is met[16]. To give agents for improved resource allocation and discover a solid solution to this problem, distributed algorithms are frequently utilized. Agent technology is a significant area of study that aids in our ability to improve user profiles and learn more about the applications in the field[17]. The main focus of distributed architecture is on multi-agent system-based techniques that improve the ability to schedule each department and avoid resource conflicts[18].

5 EXISTING SYSTEM

The construction of a timetable is tedious and time-consuming. Since there aren't any active timetable generators, this is currently done manually. The main issue while inputting the timetable is slot collisions. Slot collisions are the main issue while entering the timetable. As a result, even previously developed software does not follow the standards. So, the current system is time-consuming, a tedious

process that requires manual labor and simultaneously, with little flexibility.

The platform we used to develop this software is a web application. The programming language used to implement this software is JavaScript. In the development of the UI, React.js is used while the backend is implemented using node.js. To make it a multi-user app login facility is provided and to store the data corresponding to the user, the MongoDB database is used locally. The project is currently run on local servers but we have deployed it using Heroku and we have also changed the database from local to MongoDB atlas.

There are five major steps involved in the proposed work:

- Step 1: We have designed our algorithm for this problem by analyzing the problem deeply and designing the UI for the project.
- Step 2: To improve the efficiency of our algorithm and calculated its time complexity. We also proved the correctness of our algorithm.
- Step 3: To implement the algorithm on the backend and get the desired results by providing our dummy inputs.
- Step 4: The backend was implemented and the Rest of API's were developed to interact with the database and frontend
- Step 5: Frontend was developed and iteration of the backend with our UI is done in this step.

The entire process of creating a schedule is done manually while considering all potential limitations, both large and little.

6 PROPOSED SYSTEM

Current system generates a timetable manually. It requires lots of time. The creation of a semester schedule is one of the top duties at the beginning of each academic year. This may seem simple, but creating a schedule that takes into account everyone's availability for all semesters can be a hard job. The manual process of creating timetables in most cases can be tedious and time-consuming for faculty members.

The final system should be able to construct timetables entirely automatically, which will save an institute administration a tonne of time and work. It will provide a schedule that can be used for all semesters as well. The timetable should be planned following the university's set schedule for each course and the workload of the faculty members who will teach the corresponding disciplines. This also emphasizes the efficient use of resources, such as academic personnel, labs, and spaces. These inputs will be used to build potential timetables for the working weeks days for teaching faculty. This will integrate by using all resources as efficiently as possible while taking into account the limits.

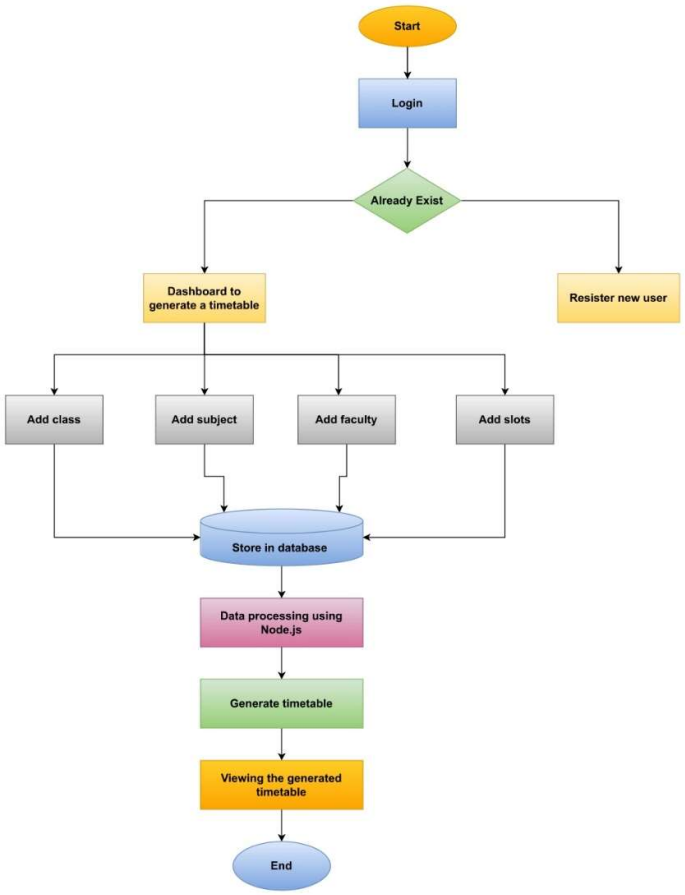

Fig. 1. Flow of Proposed System

The home page contains a Login and Registration Page, where the teacher will fill in some basic information about his or her appointment, and the section to create a teacher account in the application. You will be taken to the homepage, where you will find several sections and a menu bar with choices for adding classes, subjects, teachers, and spaces to the currently selected page. After selecting a section and filling out its details, click the "add" button to add the information to the database. If you click on the home button, you will be taken to the appropriate page. And the database has all information at all times. Check out all the information on all the professors, classes, times, and subjects. Then Once your task is complete you can exit the app. An application to connect to a website. With this website, we can

create, add, update or delete teacher information as per your concerns. We can use the website to store teacher information and thus be able to create our timetable. You will be taken to a website with sections for First, Second, and Third; after selecting one, you will be taken to a page with a faculty name and a button that, when clicked, will display a timetable. There are other details about the professors to be added when you click on "Add Faculty, Classroom, and Times," including Teacher Name, Subject, Lecture Timing, Slots, Semester, Credit, and Generate.

7.1 ALGORITHM DESCRIPTION

As the Time table-generating problem is one of the N-P hard problems, so it is difficult to get an optimal solution. The algorithmic approach which we used in this project is Heuristic Approach. All the hard constraints are dealt with by using Constraint-based Programming. The algorithm has two functions, one main function generates a timetable and a supportive function and day are used in the main function. The features and Constraints implemented can be found here.

Our algorithm takes multiple inputs which are listed below:

instances:

Data Structure which stores info about provided slots to be organized i.e.[[Ti, Ci, Si, LTi, Li],.....,[Tn, Cn, Sn, LTn, Ln]]

Here

T = Teacher
C = Class
S = Subject
LT = noon lectures
L = Labs

ins = which keeps a record of lecture assigned & will be added in the generating function according to the number of slots given

givenSlots:

The data structure which stores info about Given Slots on each day i.e.[3,4,5,3,2]

classes:

The data structure which stores info about Classes(classes) i.e.["A", "B"]

teachers:

The data structure which stores info about Teachers i.e["T1", "T2"]

Our algorithms generate time table section wise mean it handles time table of one section at a time and checks in the provided slots. If it founds a slot related to the class, it adds that instance to the section instances data structure. This was done in

the first half. In the second half, again a section is selected, and iterating through the slots on the given day and the section instance, a slot in time table is assigned. And the end of the algorithm we get an array that has nested arrays. Each nested array represents time table of a section.

7.2 PSEUDO CODE

Input:

In this algorithm, we will be giving the following input to get desired results.

instances:

Data Structure which stores info about provided slots to be organized i.e. [[Ti, Ci, Si, LTi, Li],.....,[Tn, Cn, Sn, LTn, Ln]]

Here
T = Teacher
C = Class
S = Subject
LT = noOfLectures
L = Labs
ins = which keeps a record of lectures assigned & will be added to the generating function according to the number of slots given

givenSlots:

The data structure that stores info about GivenSlots on each day i.e.[3,4,5,3,2]

classes:

Data structure that stores info about Classes(classes) i.e.["A", "B"]

teachers:

Data structure that stores info about Teachers i.e["T1", "T2"]

Variables used in the algorithm

- **sectionInstances**: data structure to store info about each section
- **TT**: data structure which is initialized with all given slots with 0 and further on the variable containing info about lecture replaces zero which is decided and given that specific slot
- **teacherTT**: it stores info about each teacher and the slot in which he is assigned a lecture
- **numOfDays**: it stores the total working days;
  Flags & Counters to keep track of clashes
- **regenerateTimeTableCountSec**
- **regenerateTimeTableFlagSec**: flag to check if there comes any clash

- **regenerateTimeTableListSec**: Keeps record of input that causes the clash
- **timeTableNotPossibleCount**: keeps count of how many time timetable generation fails on specific input
- **impossible**: it says that it is impossible to generate time table with given data

8 RESULT ANALYSIS

8.1 Dashboard

This page will be shown after a user successfully login to the application. It consists of a header and a navigation panel.

- **Home Button:** This will lead to the dashboard whenever someone presses it.
- **Logout Button:** This will log out a user from the application and takes him/her to the welcome screen. In the navigation panel, there are 5 buttons which are described below.
- **Classes:** It has a submenu that shows Add and All Classes Button. By clicking the Add button, a new page will be opened where the user can add a new class. By clicking on the All Classes button, all the classes will be shown on the UI.
- **Subjects:** It has a submenu that shows Add and All Subjects Button. By clicking the add button, a new page will be opened where the user can add a new object. By clicking on the All Subjects button, all the subjects will be shown on the UI.
- **Teachers:** It has a submenu that shows Add and All Teachers Button. By clicking the add button, a new page will be opened where you can add a new teacher. By clicking on the All Teachers button, all the teachers will be shown on the UI.
- **Slots:** It has a submenu that shows Add and All Slots Button. By clicking the Add button, a new page will be opened where you can add a new slot. By clicking on the All Slots button, all the slots will be shown on the UI.
- **Generate:** This button will send a request to the backend to run the algorithm and will show the returned output.

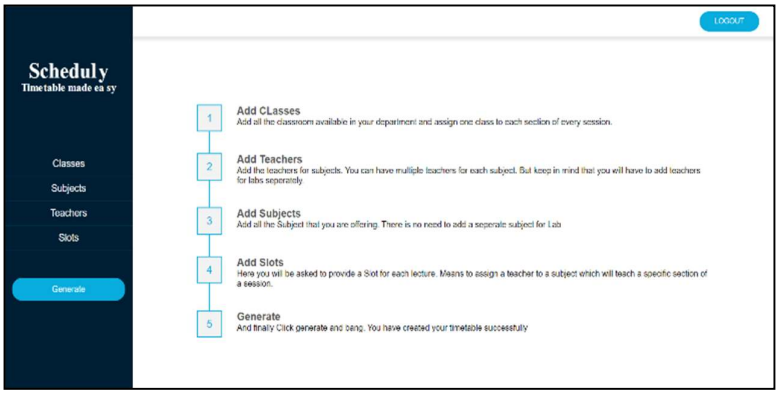

Fig. 2. Dashboard Page Interface

8.2 View All Slots

This will show all the slots added by a user. You can go to this page by clicking on slots in the navigation panel on the left and selecting all slots from the submenu. It has only one button named Remove which will remove an instance of a slot.

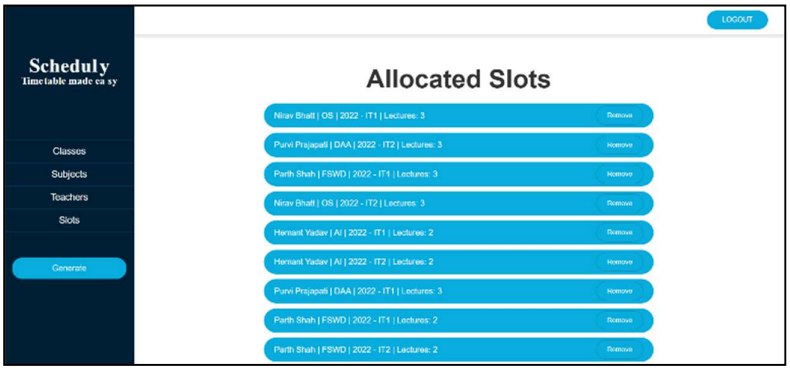

Fig. 3. View Slots/Lectures Page Interface

8.3 Timetable

By clicking on the Generate Button from the navigation panel on the left, you will see the generated timetable. Our output will be shown in tables on this page. Each table will represent time table of one class. Each row will represent a working day and each slot will represent a time slot. In each cell, we will show the assigned teacher name and assigned subject name.

2022-5IT2

Days	8am-9am	9am-10am	10am-11am	11am-12pm	1pm-2pm	2pm-3pm	3pm-4pm
Monday	DE Nirav Bhatt	EE Purvi Prajapati	SGP Nirav Bhatt	JAVA LAB Parth Shah		EE LAB Purvi Prajapati	DCN LAB Hemant Yadav
Tuesday	DCN LAB Hemant Yadav	JAVA Parth Shah			EE LAB Purvi Prajapati	HSS Nishi Purohit	JAVA LAB Parth Shah
Wednesday	EE Purvi Prajapati	DE LAB Nirav Bhatt			SGP Nirav Bhatt	JAVA Parth Shah	DCN Hemant Yadav
Thursday		DE LAB Nirav Bhatt	JAVA Parth Shah	DE Nirav Bhatt	DCN Hemant Yadav	JAVA LAB Parth Shah	HSS Nishi Purohit
Friday	EE Purvi Prajapati	HSS Nishi Purohit	DCN Hemant Yadav	DE Nirav Bhatt			

Fig. 4. Output page UI (5IT1)

2022-5IT1

Days	8am-9am	9am-10am	10am-11am	11am-12pm	1pm-2pm	2pm-3pm	3pm-4pm
Monday		HSS Nishi Purohit	EE Purvi Prajapati		DCN LAB Hemant Yadav		
Tuesday	EE Purvi Prajapati	DCN LAB Hemant Yadav	DE Nirav Bhatt	HSS Nishi Purohit	JAVA Parth Shah	DCN Hemant Yadav	DE LAB Nirav Bhatt
Wednesday	JAVA Parth Shah	DCN Hemant Yadav	SGP Nirav Bhatt	EE LAB Purvi Prajapati	JAVA LAB Parth Shah	DE LAB Nirav Bhatt	DE Nirav Bhatt
Thursday	DCN Hemant Yadav	JAVA Parth Shah		HSS Nishi Purohit	JAVA LAB Parth Shah	DE Nirav Bhatt	SGP Nirav Bhatt
Friday		JAVA LAB Parth Shah	EE Purvi Prajapati	EE LAB Purvi Prajapati			

Fig. 5. Output page UI (5IT1)

9 FUTURE SCOPE

We have presented a model for timetabling problems in this paper. The scheduling problem is considered as an optimization problem, it cannot be solved with a fixed objective function. So, following a detailed literature review, NP-hard was selected to develop the University's course schedules. This timetabling project seeks to generate near-optimal timetables using the principles of the NP-hard algorithm. It is easy to understand, has less paperwork, is effective, and is automated, which is useful for the faculty's administrators. The proposed system can only construct timetables based on a few strict constraints, and it only provides optimal solutions, not the best ones, and the NP-hard problem itself has a long execution time.

10 CONCLUSION

Managing a large faculty and giving out assignments on time is a

physically challenging undertaking. Therefore, our suggested system will aid in resolving this contradiction. As a result, we can create a schedule for any quantity of courses and semesters. With the help of this program, you may make flexible pages that can be used with a variety of tools that are more effective and liberated. This system generates different timetables for each class, genre, and lab. If another timeline is required, it can be created using a mix of various slots. You feel the pain of installing a timetable while the project minimizes time utilization. There won't be any scheduling issues because of how the project is being developed.

References

1. K. S. Shehadeh, A. E. M. Cohn, and R. Jiang, “Using stochastic programming to solve an outpatient appointment scheduling problem with random service and arrival times,” Naval Research Logistics, vol. 68, no. 1, pp. 89–111, 2021.
2. Jian, Dipti Srinivasan Tian Hou Seow, and Xin Xu. "Automated Time Table Generation Using Multiple Context Reasoning for University Modules." In 2002 IEEE conference.
3. Chowdhary, Anuja, Priyanka Kakde, Shruti Dhoke, Sonali Ingle, Rupal Rushiya, and Dinesh Gawande. "Timetable generation system." International Journal of Computer Science and Mobile Computing 3, no. 2 (2014): 410-414.
4. Nanda Nanda, Anirudha, Manisha P. Pai, and Abhijeet Gole. "An algorithm to automatically generate schedule for school lectures using a heuristic approach." International journal of machine learning and computing 2, no. 4 (2012): 492.
5. Elkhyari, Abdallah, Christelle Guéret, and Narendra Jussien. "Solving dynamic timetabling problems as dynamic resource constrained project scheduling problems using new constraint programming tools." In Practice and Theory of Automated Timetabling IV, pp. 39-59. Springer-Verlag, 2003.
6. M. D. Boomija, R. Ambika, J. Sandhiya, P. Jayashree, “Smart and Dynamic Timetable Generator”, International Journal for Research in Applied Science and Engineering Technology, March 2019.
7. V. Abhinaya, K. Sahithi, K. Akaanksha, “Online Application of Automatic Timetable generator”, International Research Journal of Engineering and Technology, February 2019.
8. Fredrikson, Rasmus, and Jonas Dahl. "A comparative study between a simulated annealing and a genetic algorithm for solving a university timetabling problem." (2016).
9. Schaerf, Andrea. "A survey of automated timetabling." Artificial intelligence review 13, no. 2 (1999): 87-127.
10. Sani, H. M., and M. M. Yabo. "Solving timetabling problems using genetic algorithm technique." International Journal of Computer Applications 134, no. 15 (2016).
11. Burke, Edmund Kieran, and Sanja Petrovic. "Recent research directions in automated timetabling." European journal of operational research 140, no. 2 (2002): 266-280.
12. Zhang, Dezhen, Saijun Guo, Weishi Zhang, and Shujuan Yan. "A novel greedy heuristic algorithm for university course timetabling problem." In Proceeding of the 11th World Congress on Intelligent Control and Automation, pp. 5303-5308. IEEE, 2014.

13. Rohit, P. S. "A Probability-Based Object-Oriented Expert System for Generating Time-Table." International Journal of Research in Computer Applications & Information Technology 1, no. 1 (2013): 52-58.
14. Asmuni, Hishammuddin, Edmund K. Burke, and Jonathan M. Garibaldi. "Fuzzy multiple heuristic ordering for course timetabling." In The Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI05), London, UK, pp. 302-309. 2005.
15. Hsu, Chin-Ming, and Hui-Mei Chao. "A Two-Stage Heuristic Based Class-Course-Faculty Assigning Model for Increasing Department-Education Performance." In 2009 International Conference on New Trends in Information and Service Science, pp. 256-263. IEEE, 2009.
16. Deris, Safaai, MohdHashim, and SitiZaiton. "Solving University Course Timetable Problem Using Hybrid Particle Swarm Optimization." pp 93-99. 2009.
17. Burke, Edmund K., J. Silva, and Eric Soubeiga. "Multi-objective hyper-heuristic approaches for space allocation and timetabling." In Metaheuristics: progress as real problem solvers, pp. 129-158. Springer, Boston, MA, 2005.
18. Soria-Alcaraz, Jorge A., et al. "Iterated local search using an add and delete hyper-heuristic for university course timetabling." Applied Soft Computing 40 (2016): 581-593.
19. Chowdhary, Anuja, Priyanka Kakde, Shruti Dhoke, Sonali Ingle, Rupal Rushiya, and Dinesh Gawande. "Timetable generation system." International Journal of Computer Science and Mobile Computing 3, no. 2 (2014): 410-414.
20. Jha, Sujit Kumar. "Exam timetabling problem using genetic algorithm." International Journal of Research in Engineering and Technology 3, no. 5 (2014): 649-654.
21. Bhatt, Nikita, Nirav Bhatt, and Purvi Prajapati. "Deep learning: a new perspective." Int. J. Eng. Technol. Manag. Appl. Sci.(IJLTEMAS) 6, no. 6 (2017): 136-140.