An Automated Timetable Generator for College

[1]K Hari Priya, [2]G Laxmi Deepthi, [3]K Bhagya Rekha, [4]N Sai Srithaja, [5]MD Abdul Rab, [6]N Venkata Bharadwaja
[1,2,3,4]Assistant Professor, [5,6]B. Tech Graduate Dept of CSE VNR VJIET, Hyderabad, India.
[1]haripriya_k@vnrvjiet.in, [2]laxmideepthi_g@vnrvjiet.in, [3]bhagyarekha_k@vnrvjiet.in, [4]saisrithaja_n@vnrvjiet.in,
[5]mohdrab01@gmail.com, [6]mohdrab01@gmail.com

Abstract **Our aim is to come up with a feasible solution for creating a schedule in a university department. We have taken up this problem since we are aware that it is a persistent problem in educational institutions. We'll use genetic algorithms to tackle a complex timetabling problem for a university or college. In order to get workable timelines in a fair amount of time, heuristics and context-based reasoning were applied. Apart from providing a feasible solution, we will also be developing a UI that will be helpful for the college administrators to set up constraints in real-time. They can add the available subjects, faculty, no. of hours, and all the required data using the UI. Then the algorithm would be run at the backend and generate the time tables which also would then be rendered in the UI.**

Keywords — Timetable, Hybrid, Genetic, Heuristic, Constraints, User Interface(UI).

I. INTRODUCTION

Colleges have different courses, faculty are dealing with different courses in different semesters and will handle multiple subjects which takes much time and have to be done without overlapping. Automatic Timetable Scheduling is our requirement since managing time effectively is important. Manually scheduling is challenging, especially when the quantity of data and resources to deal with grows, and there is a chance of collisions.

It's a complex combinatorial problem with both soft and hard constraints. University course scheduling problem is an NP hard problem, which means it cannot be solved in polynomial time as the size and complexity of the problem grows exponentially.

To provide a roadmap for this paper, the first section contains an introduction to the paper, followed by related work in section 2, proposed system in section 4, implementation in section 4, results and discussions in section 5 and in section 6 there is conclusion.

II. RELATED WORK

According to Abeer Bashab Et. Al, [5] the metaheuristic Algorithmic works better because it leverages current information gathered by algorithm to better determine which alternative solution should be evaluated next, or how the next candidate should be produced..

According to Swapnil Markal Et. Al, [4] they applied the Genetic Algorithm, which is a subset of Evolutionary Computation, a broader subject of computation. In comparison to other approaches, the clashes between the subjects are properly handled.

According to V. Abhinaya Et. Al, [7] they used a standard backtracking method, which is a brute force strategy. It takes a long time to compute and is only appropriate for one class.

According to Mei Ching Chen Et. Al, [2] meta heuristic based approaches are quite prevalent in solving the university scheduling problem, and they are closely followed by hybrid methods. Hyper-heuristic techniques, on the other hand, are also effective.

Shraddha Thakare Et. Al, [3] they proposed a genetic algorithm that was acceptable in speed and converged faster than previous basic algorithms. It should also be noted that this model can be improved by incorporating other modifications.

Mr. Mallikarjuna Nandi Et. Al, [8] proposed an optimization algorithm that incorporates several techniques to improve search efficiency. It also addresses a few crucial hard constraints, such as clashes between faculty availability, as well as non rigid soft constraints, such as search procedure optimizing objectives.

Amin Rezaeipanah Et. Al, [6] To overcome the timetabling problem, suggested a technique combining Improved PGA Algorithm and Local Search. This technique has been evaluated on numerous benchmarks, and the results suggest that it is more efficient and effective than other traditional approaches for solving UCTP.

Cheng Weng Et. Al, [12] developed a version of the method that improves global exploration by combining a global best model motivated by particle swarm optimization with the great deluge strategy to promote local exploitation. This approach strikes a good balance between exploration and exploitation.

 DOI : 10.35291/2454-9150.2022.0498

Shraddha Shinde Et. Al, [17] showed that using a genetic algorithm to schedule lectures is efficient and helpful. Using the technique they have described, they have shown a strong potential for a future leading timetable is more fairer to students. The framework is simple to use and looks to be directly applicable to a variety of other scheduling challenges.

According to H. Algethami Et. Al, [1] To solve the university scheduling challenge, they adopted a mixed integer programming technique. Faculty requirements, preferences, and availability, as well as timeslot and room assignment constraints, were all taken into account. They wanted to make the most of faculty assignments and events by customizing them to their preferences and requirements. The focus of their future study will be on heuristic solution approaches for larger TU instances.

As per David Schneider Et. Al, [9] They have tried to detect whether a given timetable is feasible or not based on the presence of binary session conflicts. Secondly, they wanted to see if the existing ProB tools could be used to run models in a production system as part of a larger project.

As per Tanzila Islam Et. Al, [11] have applied the scoring idea, in which they attempt to create the best routine based on the score. They applied the Tabu Search concept, which is an optimization problem-solving meta-heuristic. The Tabu Search method yields a sub optimal first result. Tabu Search offers a realistic answer that is completed in a reasonable amount of time.

As per Mayuri Bagul Et. Al, [14] evolutionary strategies were applied to solve the university timetabling problem. Methodologies such as evolutionary algorithms, genetic algorithms, and others have been applied, with variable results. They used a genetic algorithm to tackle it in their research. They also employed a mimetic hybrid , genetic artificial immune network algorithm to solve the problem, which they compared to the findings of the Genetic Algorithm. The genetic artificial immunity network beats the Genetic Algorithm to the best possible option.

As per Parkavi A Et. Al [10], The system will take a variety of inputs, including the quantity of subjects, teachers, their workloads, the semester, and the priority of the subject. It will produce potential time schedules for the working weeks for teaching faculty based on these inputs. This will integrate by using all resources as efficiently as possible while taking into account the limits.

As per Dipesh Mittal Et. Al [13], We are able to produce a timetable that is more exact, precise, and error-free by employing genetic algorithms, which also helps us build timetables in less time. The institute's shared mandatory classes are all included in the first phase and are organised by a central team. The distinct departmental classes are included in the second phase. Currently, this schedule is created manually by modifying those from previous years with the sole purpose of creating a schedule that is workable.

As per Mohammed Azmi Al-Betar, Et. Al [15], The combination of a population-based global search and local improvement is a core component of memetic computing. This hybridization should achieve a balance between searching and using the search space. The hybrid harmony search algorithm (HHSA), a memetic computing technique created for UCTP, is suggested in this study. The harmony search algorithm (HSA), a metaheuristic population-based technique, has been hybridised in HHSA by 1) hill climbing to enhance local exploitation, and 2) a global-best particle swarm optimization idea to enhance convergence.

As per Shengxiang Yang, Et. Al [16], in this study, local search (LS) approaches and guided search algorithms (GAs) for the UCTP are examined. On the basis of a data structure that contains information extrapolated from successful individuals of earlier generations, the guided search technique is utilised to produce children into the population. The proposed GAs' search efficiency and individual quality are enhanced by the LS methods' usage of exploitative search. On two sets of benchmark problems, the suggested GAs are put to the test in comparison to a group of cutting-edge techniques from the literature.

III. PROPOSED SYSTEM

A. Front-end – add directory-file structure
B. Backend-end
C. workflow
D. Integration-front-back - architecture

A. Building Front End

ReactJs was used to build the frontend of our project. The hierarchy of the components used to develop user interfaces is depicted in the diagram

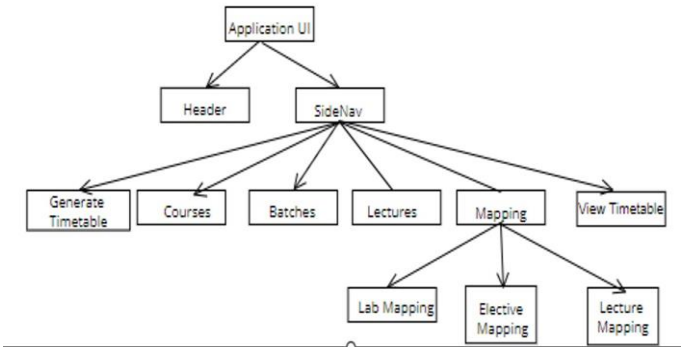

Fig. 1. Component Hierarchy

As shown in the fig. 1, there are two main components for theReact App:

- Header - which displays the title name and sidenav logo

 DOI : 10.35291/2454-9150.2022.0498

- SideNav -It is further subdivided into 6 other components as mentioned below. Its sub-components are:
  - Generating Timetable - The component contains a button which on click generates a new time table
  - Courses - This Component is responsible for adding the courses in the database
  - Batches - This component is responsible for adding the batches in the database
  - Lectures - This Component is responsible for adding the professors data in the database
  - View Timetable - This Component is responsible for rendering the Time table.
  - Mapping - It is further subdivided into 3 other components as mentioned below. Its sub-components are:
    - Lab Mapping - This component is responsible for mapping the Labs with courses and batches in database
    - Electives Mapping - This component is responsible for mapping the Electives with courses and batches in database
  - Lectures Mapping - This component is responsible for a mapping the Lectures with courses and batches in database

B. Building Back -end

We developed a Back-end using python which includes several methods, and models for several specific functionalities. Below are a few modules and their descriptions we used in our backend.

- **Router Module** – which contains all the routes for getting data from the front end and also for sending back data to the front end from the database.
- **Time Table Module** – which consists of the methods to create gene, genome and perform operations like crossover, detecting collisions in time table.
- **Scheduling Module** – which is responsible for using time table module methods.It is also involved in scheduling and rescheduling the time table.
- **Model Related Module** –Which consists of all the schemas of different types of data that is utilized by the scheduler module to generate a time table.
- **Data Module** – This modules consists of methods to to access the database to provide the data to scheduler module

C. Workflow to Generate a Time Table

The workflow is depicted in fig 2 and the process begins with gathering input from components such as batches, professors, and courses.Each user input is stored in the database using API calls.The next step in this process is to map the input, which involves assigning professors to courses in each batch.Once the mapping is complete, we send the mapping data to the backend via API.

The backend router module listens for and responds to front end requests.The scheduler module in the backend creates genomes or individual time tables using the mapped data.The number of genomes produced is proportional to the population size.The next step in this procedure is to examine the number of collisions in the generated genomes.

If there are time table collisions, we promote 10% of time tables, crossover and mutate the remaining 90% of time tables to create new genomes or new generations of time tables, and if we don't get a time table without collisions within a limited number of new generations, we re-schedule so that scheduler module re- initialize the mapped data to repeat previous step.

This step is repeated until we have a time table with no collisions.

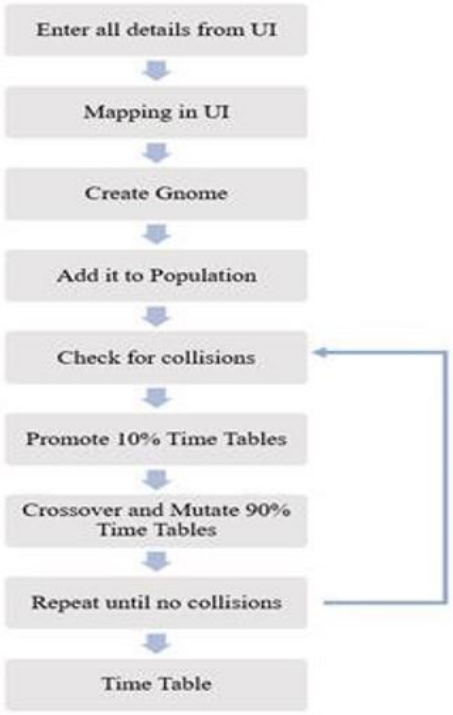

Fig. 2. Workflow

The steps shown in above figure 2. gives a brief idea of the methodology being followed in the existing system for timetable generation. Let us understand it some more elaboratively.

 DOI : 10.35291/2454-9150.2022.0498

a. Enter all details from UI – This step involves entering of the batch, course and faculty details from UI. We can also delete any entry at any point of time if we have entered wrong details.

b. Mapping in UI – After entering all the details in the previous step, now we must create mappings for lectures, labs and elective courses with specific batch and faculty. We have seperate mappings screen for that, wherein user can easily create these mappings.

c. Create Gnome – These mappings data would be stored in DB. At backend, we make use of these mappings to generate a master data, that would be responsible to create the timetable. As part of it, the first step would be to create gnome. Gnome here is nothing but generation of the required time slots and allot the mapped data accordingly into those slots randomly.

d. Add it to the population – A certain number of gnomes (timetables) would be created and added to population list.

e. Check for collisions – Now, all the timetables in the population list would be checked for the collisions. We here check for 3 types of collisions specific to the type of mapping. They are – Batch collision, Faculty Collision and Room Collisions.

f. Promote 10% timetables – Out of the population of timetables, based on collisions, we promote 10% of timetables for the next step.

g. Crossover and Mutate 90% of the timetables – The rest of 90% timetables in population list, are then added to balance filler and we apply genetic operations on them like crossover and mutation so that we hope to remove the collisions from these timetables and generate new population.

h. Repeat until no collisions – If we get a timetable with zero collisions, then we send it to the UI. But if we do not get any timetable with zero collisions even after certain amount of waiting time (say 10000 iterations), then we re-schedule the algorithm from start.

i. Timetable – When we get a timetable with zero collisions at backend, we send it as an object to the UI. And the timetables for all batches are displayed at user screen

D. Connecting front-end to the back-end

This is another critical component of the application. We send a GET call to a backend API whenever a user clicks the Timetable Generate button in the UI. The backend retrieves the necessary data from the database that we previously saved, the scheduler method creates the genomes with necessary information, these genomes undergo repeated crossover and mutation to generates the timetable with no collisions, and generated time table is sent as the response to the frontend

At last, Frontend process the response received from the backend and displays it to the users in an understandable way.

IV. IMPLEMENTATION

The front-end module screens where users enter input for Batches, Courses, and Professors, as well as mapping screens for lectures, labs, and electives that use the user input provided in previous screens, are shown below.

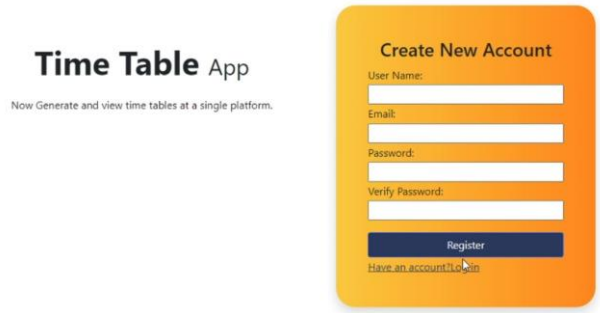

Fig. 3. Creating New Account

Fig. 4. Login Page

Fig. 5. Home Page

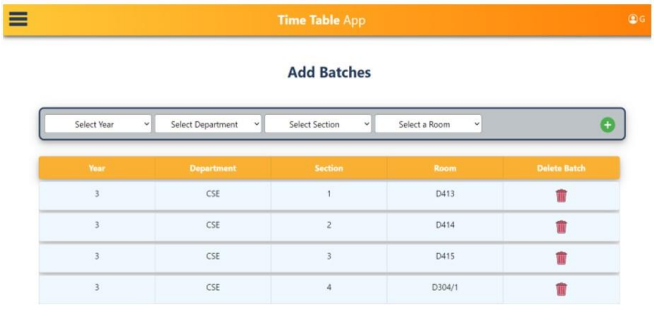

Fig. 6. Batches Screen

 DOI : 10.35291/2454-9150.2022.0498

Fig. 7. Courses Screen

Fig. 8. Professors Screen

Fig. 9. Lectures Mapping Screen

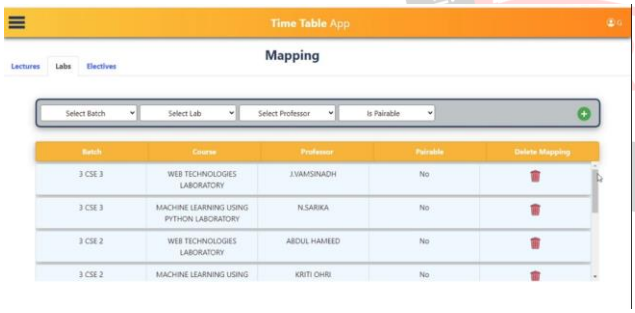

Fig. 10. Labs Mapping Screen

Fig. 11. Electives Mapping Screen

V. RESULTS AND DISCUSSION

A. *Output Screens*

We have included a generate button in the Generate Time Table Component to generate time tables. When we click the generate button request is sent from the front-end to thebackend router module to start generating timetables.

After successful timetable generation, the router module sends this output as the response to the front end, which displays the generated timetables as shown below.

We can view different timetables generated for different batches by selecting the batches from the dropdown menu asshown in fig. 11

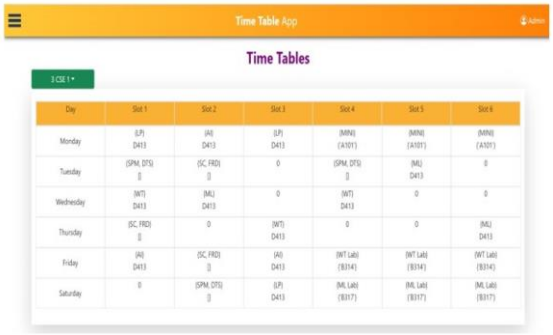

Fig. 11. Generating Time Table Screen

B. *React App Testing*

```
PASS src/Testing/Batch.test.js
  Testing Batch Component
    √ 1.Renders Add Batches Text (157 ms)
    √ 2.Rendering placeholders correctly (30 ms)
    √ 3.Rendering add icon (80 ms)
    √ 4.Rendering delete icon (79 ms)

PASS src/Testing/Navbar.test.js
  Testing Navbar and side Menu
    √ 1.Rendering Time Table App Text (244 ms)
    √ 2.Rendering User Icon (90 ms)
    √ 3.Rendering Side Menu (38 ms)

PASS src/Testing/Course.test.js
  Testing Course Component
    √ 1.Renders Add Courses Text (195 ms)
    √ 2.Rendering placeholders correctly (25 ms)
    √ 3.Rendering add icon (119 ms)
    √ 4.Rendering delete icon (53 ms)
    √ 5.Testing Input Room No Disabled (28 ms)
    √ 6.Testing Course Type to enable/disable room no field (37 ms)

PASS src/Testing/Mapping.test.js
  Testing Mapping Component
    √ 1.Renders Mapping Text (226 ms)
    √ 2.Rendering placeholders correctly (79 ms)
    √ 3.Rendering add icon (178 ms)

PASS src/Testing/Lecturers.test.js
  Testing Lecture Component
    √ 1.Renders Add Batches Text (12 ms)
    √ 2.Rendering placeholders correctly (7 ms)
    √ 3.Rendering add icon (31 ms)
    √ 4.Rendering delete icon (18 ms)

Test Suites: 5 passed, 5 total
Tests:       20 passed, 20 total
Snapshots:   0 total
Time:        3.845 s
Ran all test suites related to changed files.
```

Fig. 12. React app unit test report

The above fig. 12 is the unit test report of the React App(Frontend).

 DOI : 10.35291/2454-9150.2022.0498

C. *Flask App Testing*

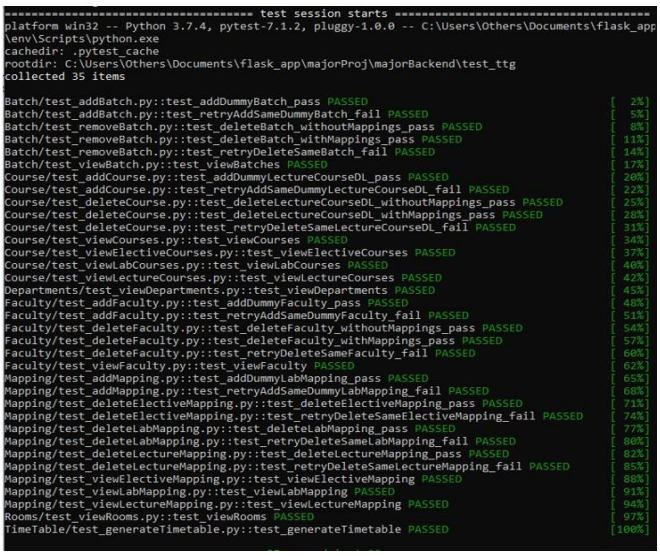

Fig. 13. Flask app unit test report

The above fig. 13 is the unit test report of the Flask App(Backend).

VI. CONCLUSION

According to a large number of papers in the scientific literature, the university course scheduling problem is an active and essential research topic. The initial appearance of international competitions in timetabling continues to inspire. This research looks at the different strategies presented in the last 10 years to handle the challenge of university course scheduling (both real world and benchmark). The approaches are divided into various groups. They are also organized chronologically by year of publication. In addition, the limitations of each methodological group are discussed in this study. We have used the genetic algorithm approach for timetable generation as part of this project. As a result, we could generate timetables very efficiently using this approach. We were able to handle the collisions like batch collisions, room collisions and faculty collisions for the courses to be scheduled. The algorithm gave very promising results.

We can include below mentioned features as a future-scope of the project -

- It can be extended to different departments in the college.
- We should be able to handle elective courses in a college-specific way i.e, the elective course mappings.
- We can add the additional feature to edit the details in the UI.
- We can include the warnings to be displayed to the user for such mappings data for which timetable cannot be generated with specific reason for the better user experience.
- It can also contain the scope to swap the allotted time slots after generating the timetable.
- We can also have the faculty timetable generated accordingly.

REFERENCES

[1] H. Algethami, W. Laesanklang A Mathematical Model for Course Timetabling Problem With Faculty-Course Assignment Constraints August 2021 DOI:10.1109/ACCESS.2021.3103495

[2] MEI CHING CHEN, SAN NAH SZE, SAY LENG GOH, NASSER R. SABAR, GRAHAM KENDALL. A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities July 2021 DOI: 10.1109/ACCESS.2021.3100613

[3] Shraddha Thakare, Tejal Nikam, Prof. Mamta Patil. Automated Timetable Generation using Genetic Algorithm IJERTV9IS07056 Volume 09, Issue 07 (July 2020) IJERT

[4] Swapnil Markal, Surekha Ghorpade, Diksha Chalke. Timetable Generator e-ISSN: 2278-0661,p-ISSN: 2278- 8727, Volume 22, Issue 2, Ser. II (Mar - Apr 2020), PP 29-

[5] Abeer Bashab, Ashraf Osman Ibrahim, Eltayeb E. Abed Elgabar, Mohd Arfian Ismail, Abubakar Elsafi, Ali Ahmed, Ajith Abraham. A systematic mapping study on solving university timetabling problems using meta- heuristic algorithms. Neural Computing and Applications (2020) 32:17397–17432

[6] Amin Rezaeipanah, Samaneh Sechin Matoori, Gholamreza Ahmadi. A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Applied Intelligence volume 51, pages467–492 (2021).

[7] V. Abhinaya, K. Sahithi, K.Akaanksha. Online Application of Automatic Time-Table Generator. Volume: 06 Issue: 02 | Feb 2019

[8] Mr.Mallikarjuna Nandi, Ms.R.Priyadharshini, Ms.R.Aishwarya and Ms.M.Nandhini. AUTOMATIC TIME TABLE GENERATION. Vol 12 Issue 1 2019

[9] David Schneider, Michael Leuschel , Tobias Witt. Model-based problem solving for university timetable validation and improvement. Formal Aspects of Computing volume 30, pages545–569 (2018)

[10] Parkavi A. A STUDY ON AUTOMATIC TIMETABLE GENERATOR MAY 2018 ISSUE VOLUME 5 ISBN No. 978-81-923607-3-7

 DOI : 10.35291/2454-9150.2022.0498

[11] Tanzila Islam, Zunayed Shahriar, Mohammad Anower Perves, Monirul Hasan. University Timetable Generator Using Tabu Search DOI: 10.4236/jcc.2016.416003

[12] Cheng Weng Fong, Hishammuddin Asmuni, and Barry McCollum. A Hybrid Swarm-Based Approach to University Timetabling. IEEE Transactions on Evolutionary Computation (Volume: 19, Issue: 6, Dec. 2015) DOI: 10.1109/TEVC.2015.2411741

[13] Dipesh Mittal, Hiral Doshi, Mohammed Sunasra , Renuka Nagpure. Automatic Timetable Generation using Genetic Algorithm Vol. 4, Issue 2, February 2015. DOI 10.17148/IJARCCE.2015.4254

[14] Mayuri Bagul, Sunil Chaudhari, Sunita Nagare, Pushkar Patil. A Novel Approach for Automatic Timetable Generation. Volume 127 – No.10, October 2015

[15] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, and Munir Zaman. University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm. IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews) 42(5):664-681 DOI:10.1109/TSMCC.2011.2174356

[16] Shengxiang Yang, and Sadaf Naseem Jat. Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling. IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews) 41(1):93 - 106

[17] Mittal, Dipesh, et al. "Automatic timetable generation using genetic algorithm." *International Journal of Advanced Research in Computer and Communication Engineering* 4.2 (2015): 245-248.

 DOI : 10.35291/2454-9150.2022.0498