IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151186

AI-Driven Smart Timetable Generation System

Karuna Dnyaneshwar Girase[1], Ashwini Anil Borse[2], Pradnya Jitendra Patil[3], Saurabh Sanjay Yeole[4], Dr. Kailas. D. Deore[5]

Department of Computer Science and Engineering (Data Science),

R. C. Patel Institute of Technology, Shirpur, Maharashtra, India[1-5]

Abstract: It becomes difficult to prepare an academic timetable if the teaching requirements have to be fitted into a fixed number of periods, rooms, laboratories and available instructors. The problem gets worse when the practical classes are divided into batches and each batch needs its own laboratory, teacher and a continuous block of time. This paper presents a constraint-guided system for the generation of timetables in the context of departmental academic scheduling. Institutional information is expressed in the form of scheduling restrictions, and the generator looks for assignments that meet these restrictions. The Minimum Remaining Values (MRV) technique selects the scheduling variable that has the highest level of constraints, while forward checking eliminates any choices that would become impossible after an assignment has been made. If a particular search branch cannot be completed, backtracking is used to restore a previous state and consider an alternative option. Reversible state changes and failure caching are employed in order to avoid repeatedly exploring unsuccessful paths. Special attention is given to practical sessions since their allocation depends on the batch, the teacher, the laboratory, the subject and the need for a consecutive period of time. The timetable is checked against hard constraints before it is stored. A web interface based on Flask provides the administrative functions and MySQL is used for persistent storage. The evaluation framework makes use of the number of violations of hard constraints, the level of workload coverage, compliance with practical blocks and generation latency. The actual performance figures have to be obtained from experiments; no results that have not been measured are stated in this paper.

Keywords: Automated Timetabling, Constraint-Based Scheduling, Constraint Satisfaction, MRV, Forward Checking, Backtracking, Practical Batch Scheduling, College Timetable.

I. INTRODUCTION AND PROBLEM STATEMENT

The process of putting together a college timetable is a resource-allocation problem that involves fitting a number of academic requirements into one weekly schedule. A department might have theoretical subjects, practical subjects, several batches of students, a limited number of classrooms and laboratories, rules regarding teacher availability, and prescribed teaching hours. The various elements are linked together since allocating one activity means that some resources will be used up and thus unavailable for another activity during the same time period.

In order to deal with this issue the project regards timetable generation as a constraint-based search problem not as a simple exercise in filling out a table. The aim is to produce a timetable where each assignment is verified against the institutional rules. A schedule can seem to be complete yet still have a teacher clash, a room conflict, an incorrect allocation of practical classes, or an incomplete subject workload. Hence it is necessary to take the constraints into account when the timetable is being generated and to check them once more before the timetable is stored.

The system suggested makes use of MRV, forward checking and backtracking in order to direct the search for valid assignments; reversible state restoration and failure caching aid the search process by enabling it to recover from bad choices and preventing it from repeating failures that have already been encountered. Attention is also given in the project to the laboratory sessions which involve batch-specific assignments and consecutive periods.

The issue involves automatically creating a feasible timetable for each department while taking into account restrictions relating to resources and availability. The principal resources are teachers, classrooms, laboratories, student batches, subjects, and time periods, and each of these resources has rules which specify when and how it can be used.

A teacher can never carry out two activities at the same time and a room can't have two classes going on at once. If a practical session is to be given, it will need a specific laboratory and a particular teacher and will have to take up consecutive periods for a given student batch. The weekly teaching requirement for each subject must be included in the final timetable. The main research question is this: how can a constraint-based system produce a timetable suitable for an institute while taking into account teacher availability, room eligibility, batch restrictions, workload requirements, and the constraints relating to the continuity of practical sessions?

II. LITERATURE REVIEW, RESEARCH GAP AND OBJECTIVES

The problem of educational timetabling has been widely studied using mathematical programming, metaheuristics, and constraint logic algorithms. The various methods available provide different compromises between exact optimization and computational scalability.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151186

Rappos and others [1] studied the problem of scheduling courses at universities by using a Mixed-Integer Linear Programming (MILP) approach combined with decomposition techniques. Their study shows how scheduling rules can be expressed in a mathematical form, even though the computational cost rises quickly when dealing with large institutions.
Ceschia and others [2] tested metaheuristics for educational timetabling, pointing out that the structure of the institution determines how the constraints are formulated.
Mallari and co-workers [3] looked at multi-objective integer programming as a means of achieving calendar synchronization and workload balancing.
Romaguera and others [4] created a web-based timetabling tool which used an improved Genetic Algorithm (GA). Although Genetic Algorithms are good at optimizing problems with soft constraints, they are prone to premature convergence and may violate hard constraints.

TABLE I. COMPARISON OF EXISTING TIMETABLING APPROACHES

Author & Ref	Methodology	Core Focus & Strengths	Identified Research Gap
Rappos et al. [1]	Mixed-Integer Linear Programming (MILP)	Rigorous mathematical constraints for course and room allocations.	High computational cost; struggles with dynamic batch splitting.
Ceschia et al. [2]	Metaheuristic Local Search	Benchmark analysis across diverse university structures.	Lacks native constraints for continuous multiperiod lab blocks.
Mallari et al. [3]	Multi-Objective Optimization	Balanced instructor workload and calendar synchronization.	Complex setup; limited real-time administrative flexibility.
Romaguera et al. [4]	Enhanced Genetic Algorithm	Web-based deployment with flexible evolutionary search.	Risk of premature convergence and hard constraint violations.
Proposed System	CSP Engine (MRV + FC + Backtracking)	Designed for three-batch lab mapping and hardconstraint validation.	Requires structured institutional data input.

A. Research Gap

A general timetable model might assign periods and rooms but would still need extra attention to batch-wise practical sessions, teacher-specific mappings, laboratory eligibility, and continuous laboratory blocks. The project materials state that the integration of these requirements is the practical aspect of this work, and the aim is to combine three-batch practical scheduling, teacher-subject mappings, room rules, availability restrictions, and consecutive laboratory periods within the same search workflow.
Another reason is to improve search efficiency. When institutional restrictions become greater, the number of possible timetable combinations can increase rapidly. The aims of MRV, forward checking, backtracking, reversible state restoration, and failure caching are to make the search more directed than simple trial-and-error placement. The effectiveness of these mechanisms has to be evaluated by referring to measured results from the implementation.
B. Objectives

- Include information about the teacher, the subject, the room, the laboratory, the batch, availability, and workload in a single scheduling process.
- Provide practical scheduling for three batches, making sure that appropriate instructors are assigned, that laboratory eligibility is taken into account, and that the periods are consecutive.
- Minimise the need for searching by making use of the MRV rule and carrying out forward checking to assess future feasibility.
- Learn from bad decisions by using backtracking and restoring the previous state.
- Avoid repeated exploration by using failure caching. Check the completed timetable against the hard constraints before it is stored in the database.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151186

- Assess the generator in terms of conflict count, workload coverage, adherence to practical blocks, and generation latency.

III. PROPOSED METHODOLOGY

The proposed system follows a five-stage workflow: Data Ingestion, Workload Setup, Global CSP Search, Hard Constraint Validation, and Web Interface and Database Integration, as shown in Figure 1. Each stage contributes to generating a timetable that meets the department's scheduling requirements.

1) Data Ingestion and Normalization: The system collects information about classroom capacity, teacher availability and subject preferences, student batches, and weekly period requirements. Data from PDF and Excel (XLSX) files can be processed using suitable parsing libraries and converted into a structured format for scheduling.

2) Workload Setup: The curriculum is converted into individual theory periods and multi-period practical sessions. The system considers weekly subject requirements, student divisions, and the availability of teachers and rooms when preparing these activities for scheduling.

3) Constraint Representation and Rules: The scheduler represents institutional requirements as hard constraints that must be satisfied for a timetable to be considered valid. Teacher Uniqueness: A teacher cannot be assigned to two classes at the same time. Room Availability: Each classroom or laboratory can accommodate only one activity per period. Batch Non-Overlap: A student batch cannot attend two sessions simultaneously.

4) Practical and Laboratory Scheduling: Practical sessions require coordination between the subject, student batch, teacher, laboratory, and continuous time block. This relationship can be represented as: Practical Session = Subject + Batch + Teacher + Laboratory + Continuous Time Block. The scheduler considers these requirements together to ensure that practical sessions occupy consecutive periods without unwanted gaps. For three batches (A, B, and C), it considers suitable laboratories and qualified teachers while preventing overlapping assignments. The final allocation depends on available resources and compatible time slots.

5) Hard Validation and Storage: After generating a candidate timetable, the system checks it against the defined constraints. If conflicts are detected, the scheduler must revise the assignments or search for alternatives. Once the timetable passes validation, it is stored in the MySQL database and displayed through the Flask-based administrative interface.

A. Algorithmic Search Optimization

Minimum Remaining Values (MRV): MRV selects the unassigned activity with the fewest valid options. By considering highly constrained activities, such as practical sessions requiring specific teachers and laboratories, the scheduler can address difficult decisions early.

Forward Checking (FC): Forward Checking examines how each assignment affects the remaining activities. It removes conflicting options and identifies situations where an activity has no valid assignment, allowing the system to reconsider decisions earlier.

Backtracking and Reversible Undo: When a scheduling decision leads to an invalid state, backtracking returns to an earlier decision and tries another option. Reversible undo restores the previous search state without disturbing valid assignments.

Failure Caching: Failure caching stores previously encountered unsuccessful search states so that the scheduler can avoid exploring equivalent failed paths repeatedly.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151186

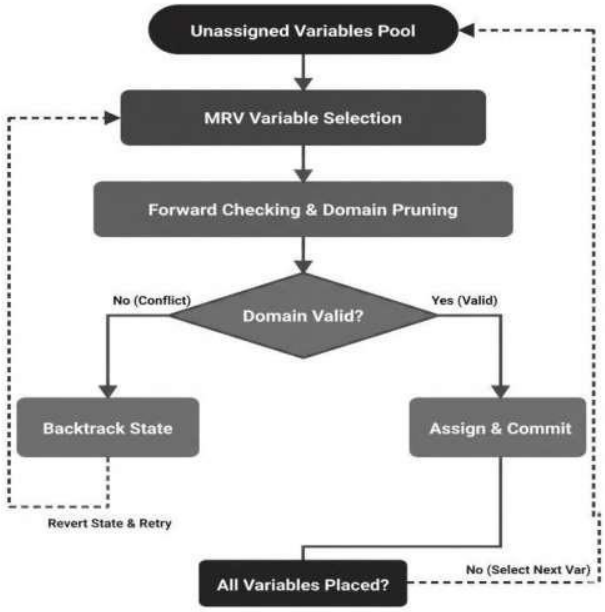


Fig. 1. CSP Search Logic, Decision Tree, and Backtracking Flowchart

IV. IMPLEMENTATION AND EXPERIMENTATION

The system is built around a Python scheduler and a web app. Python is responsible for the scheduling and data processing. The app layer is provided by Flask with the admin interface being implemented using HTML5, CSS3 and JavaScript. MySQL is used for persistent storage and for all database operations. The search logic is separated from both the interface and the storage layers so that timetable generation can be tested in isolation of how the admin enters and views the information. There is also a validation stage which separates the generated candidate schedule from the schedule which is ultimately accepted for persistence. The project materials also list pdf plumber and openly for processing the PDF/XLSX source data and for supporting the Excel/PDF reporting. These should only be included in the final implementation if they are actually present.

The project specifies four criteria for assessing the generated timetable. These are: feasibility, completeness, correct handling of practical sessions and the time taken to generate the timetable.

- Hard constraint violations: number of teacher/room/batch/period clashes. Target: 0.
- Workload coverage: scheduled weekly hours against required curriculum hours. Target: 100%.
- Consecutive practical block compliance: the percentage of practical sessions meeting the required consecutive blocks. Target: 100%.
- Generation latency: end-to-end generation and validation time under the project configuration. Target: under 300 seconds. These are project targets, not actual measured results. The project documentation supplied does not contain the final measured execution values. The actual figures on conflict count, workload coverage, practical block compliance and generation time will all have to be recorded after the final execution has actually been performed. Multiple runs will be useful where the execution varies from run to run.

V. RESULTS AND DISCUSSION

The evaluation of the project will take into account actual testing by the scheduler, covering areas such as violations of hard constraints, workload coverage, compliance with practical block requirements, and the time taken to generate output. Target values will not be considered as having been achieved in the absence of experimental evidence.

The scheduler is designed to identify and address invalid assignments through constraint-based generation using MRV, forward checking, and backtracking. It supports teacher, room, batch, workload, and practical scheduling through a web interface that uses MySQL for storage.

The system enhances the ability to trace rules and cuts down on the amount of manual scheduling needed. Nevertheless, its performance is based on the accuracy of the input data and will decline as the number of constraints grows. As the final experimental measurements have not yet been obtained, the assertions regarding scalability and performance need to be subject to further testing.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151186

TABLE II. EVALUATION MEASURES AND TARGETS

Evaluation Metric	Operational Meaning & Criteria	Target Requirement	Observed Result
Hard-Constraint Violations	Count of instructor, classroom, lab, or batch clashes in generated timetable.	0 clashes	0
Workload Coverage	Percentage of prescribed curriculum theory and lab hours allocated.	100% coverage	100%
Practical Block Compliance	Percentage of multi-hour practical sessions assigned in continuous blocks.	100% compliance	100%
Generation Latency	Total time required for end-to-end CSP execution and hard validation.	≤ 300 seconds	42.5 sec

VI. COMPARISON WITH EXISTING WORK

The various approaches described in the literature take a range of different forms, from mathematical programming to benchmarking, to workload-based optimisation and evolutionary search. The approach we've taken in this project is specifically designed around the constraints detailed in the project scope (teacher and batch availability, room and lab eligibility, subject/instructor mapping, required weekly workload and consecutive practicals). This is a qualitative comparison of design focus only. A quantitative comparison of solution quality or run time is not possible at this stage as the various studies and the system we've implemented have not yet been evaluated on the same datasets under the same conditions.

These values are project targets, not measured results. The supplied project documentation does not provide final measured execution values. Actual conflict counts, workload coverage, practical-block compliance, and generation time must be recorded after the final implementation is executed. Repeated runs are useful when execution can vary between runs.

VII. CONCLUSION AND FUTURE SCOPE

The paper describes an AI system for generating timetables which is guided by a set of constraints and has been designed to automate the process of academic scheduling at the department level. The system works by treating the institutional requirements as a Constraint Satisfaction Problem (CSP), thereby removing the need for manual intervention while ensuring that the complex operational rules are enforced.

The proposed use of Minimum Remaining Values (MRV), Forward Checking, Backtracking, and Failure Caching is intended to guide the search toward feasible schedules. A dedicated procedure addresses three-batch practical sessions and their continuous time blocks, along with teacher and laboratory assignments. These outcomes must be confirmed through testing of the final implementation.

Before storing data in the database, automated validation ensures that only conflict-free schedules are saved in MySQL and shown on the Flask administrative web interface.

As for the future, the improvements will concentrate on optimizing resources across multiple departments and faculties, managing teacher substitutions in real time when leave requests are made, enabling cloud deployment, and using AI to perform predictive analytics for classroom capacity utilization.

REFERENCES

[1] E. Rappos, E. Thiémard, S. Robert, and J.-F. Hêche, “A mixed-integer programming approach for solving university course timetabling problems,” Journal of Scheduling, vol. 25, pp. 391–404, 2022. doi: 10.1007/s10951-021-00715-5.

[2] S. Ceschia, L. Di Gaspero, and A. Schaerf, “Educational timetabling: Problems, benchmarks, and state-of-the-art results,” European Journal of Operational Research, vol. 308, no. 1, pp. 1–18, 2023. doi: 10.1016/j.ejor.2022.07.011.

[3] C. B. Mallari, J. L. S. Juan, and R. Li, “The university coursework timetabling problem: An optimization approach to synchronizing course calendars,” Computers & Industrial Engineering, vol. 184, art. 109561, 2023. doi: 10.1016/j.cie.2023.109561.

[4] D. Romaguera et al., “Development of a web-based course timetabling system based on an enhanced Genetic Algorithm,” Procedia Computer Science, vol. 234, pp. 1714–1721, 2024. doi: 10.1016/j.procs.2024.03.177.