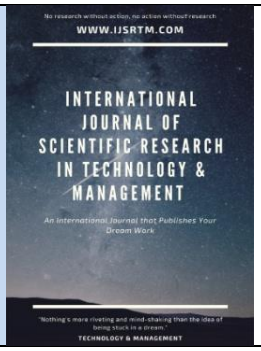
IJSRTM
https://www.ijsrtm.com
Vol. 6 Issue 1 January 2026: 22-26
Published online 14 Jan 2026
E-ISSN: 2583-7141

International Journal of Scientific Research in Technology & Management



CampusSmart Scheduler: AI-Based Automated Timetable Management System

Jagrati Agrawal
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
jagratiagrawal052@gmail.com

Medha Agrawal
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
medhaagrawal2525@gmail.com

Narayani Puranik
Dept. of Computer Science Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
puraniknarayani014@gmail.com

Maneshwari Pawar
Dept. of Computer Science and Engineering
Oriental Institute of Science and Technology
Bhopal, Madhya Pradesh, India
maneshwaripawar@gmail.com

Abstract*— Designing an academic timetable is a complex and time-consuming task that requires balancing multiple constraints related to classrooms, faculty availability, student batches, and institutional policies. This challenge, formally known as the University Course Timetabling Problem (UCTP), belongs to the class of NPhard combinatorial optimization problems due to its vast search space and tightly coupled constraints. In most universities, timetable preparation is still carried out manually, often taking 12–15 days and resulting in poor utilization of physical resources, typically below 40%. This paper presents Campus Smart Scheduler, an intelligent and automated timetable management system developed using the OptaPlannerconstraint-solving framework. The proposed solution models the UCTP as a Weighted Constraint Satisfaction Problem (WCSP) and employs a hybrid approach that combines Constraint Satisfaction Programming for feasibility with Genetic Algorithms for optimization. A key contribution of this work is a mathematically defined softconstraint model that explicitly promotes smart space utilization by penalizing room underutilization and fragmented scheduling. Experimental results demonstrate significant improvements in scheduling speed, feasibility, and room utilization efficiency, making the system a practical and scalable solution for modern academic institutions.

Keywords*— University Timetabling, Constraint Satisfaction Problem, Genetic Algorithm, OptaPlanner, Smart Space Utilisation, NP-Hard Problem.

I. INTRODUCTION

Timetabling in higher education institutions involves assigning courses to time slots, classrooms, and instructors while satisfying a wide range of constraints. These constraints include avoiding scheduling conflicts, respecting faculty availability, and ensuring that classroom capacities are sufficient for enrolled students. This problem, commonly referred to as the University Course Timetabling Problem (UCTP), is well known for its computational complexity and has been classified as NP-hard. Despite advancements in computing, many institutions continue to rely on manual or semi-automated scheduling methods. Such approaches are time-consuming and often lead to suboptimal outcomes, including underutilised classrooms and fragmented schedules. As institutions expand and resources are shared across departments, the limitations of manual scheduling become more evident.

The motivation behind this work is to design a system that not only generates feasible timetables but also focuses on optimising resource usage, particularly classroom space. The proposed CampusSmart Scheduler aims to reduce administrative effort while delivering high-quality schedules that are suitable for real academic environments.

II. RELATED WORK

The University Course Timetabling Problem has attracted significant attention from researchers due to its practical importance and inherent complexity. Early approaches relied on manual scheduling or simple rule-based systems, which were suitable only for small institutions but failed to scale effectively (Carter and Laporte, 1998).

Genetic Algorithms have been widely applied to timetabling problems because of their ability to explore large solution spaces. Smith et al. (2019) demonstrated that evolutionary techniques can reduce scheduling conflicts; however, their approach primarily focused on feasibility rather than optimisation of physical resources. Gupta and Rao (2020) proposed a faculty-centric genetic scheduling model, but their results showed increased computation time as the number of constraints grew.

Constraint Programming methods model timetabling as a constraint satisfaction problem, ensuring that hard constraints are strictly enforced. Burke et al. (2021) reported that while constraint programming guarantees feasible solutions, optimisation objectives such as room utilisation are often difficult to balance effectively. Hybrid approaches combining constraint satisfaction with heuristic optimisation have therefore been proposed to address these limitations.

Recent studies have explored practical optimisation frameworks such as OptaPlanner, which supports hybrid solving techniques and is suitable for real-world deployment. Ahmed et al. (2022) used a hybrid constraint-heuristic approach to improve timetable compactness but did not explicitly address room underutilisation. This gap motivates the present work, which places resource utilisation as a primary optimisation objective.

A. Research Gaps Identified

From the literature survey, the following research gaps are identified:

a. Lack of production ready hybrid AI systems combining CSP, GA and PSO effectively.

b. Limited support for real-time dynamic editing and partial re-optimization.

c. Inadequate focus on resource utilization metrics such as room occupancy and balanced faculty workload.

d. Poor ERP and cloud integration in existing research prototypes.

e. Absence of standardized performance benchmarks for large scale academic timetables.

Table I: Comparison with Existing Approaches

Method	Scheduling Time	Room Utilisation	Conflict Handling
Manual Scheduling	Several days	< 40%	Partial
GA-Based Method	120–300 s	~70%	Yes
CP-Based Method	60–180 s	~75%	Yes
Proposed System	**5–60 s**	**90–95%**	**Yes**

III. PROPOSED METHODOLOGY

The proposed system models the timetabling problem as a weighted constraint satisfaction problem. Constraints are categorised into hard and soft constraints. Hard constraints must be satisfied under all circumstances and include conditions such as avoiding time clashes for faculty members and ensuring that room capacity is not exceeded. Any violation of a hard constraint renders a solution infeasible.

Soft constraints are used to improve the quality of feasible solutions. These include minimising idle gaps in schedules and maximising classroom utilisation. Each soft constraint is assigned a weight that reflects its relative importance. The optimisation process seeks to minimise the total weighted penalty while maintaining feasibility.

The system uses the OptaPlanner optimisation engine, which applies meta-heuristic techniques to efficiently search the solution space. This approach allows the system to generate high-quality timetables within a practical time frame, even for large datasets.

A. System Architecture

The CampusSmart Scheduler follows a three-layer architecture consisting of a presentation layer, an application layer, and a data layer. The presentation layer provides an interface for administrators to enter institutional data and view generated timetables. The application layer contains the optimisation logic and constraint definitions implemented using Java and OptaPlanner. The data layer stores information related to courses, faculty, rooms, and time slots in a relational database. Data entered by the administrator is transformed into planning entities and passed to the solver. Once an optimised timetable is generated, the solution is stored in the database and presented through the user interface. This modular design improves scalability and allows the system to be adapted to different institutional requirements.

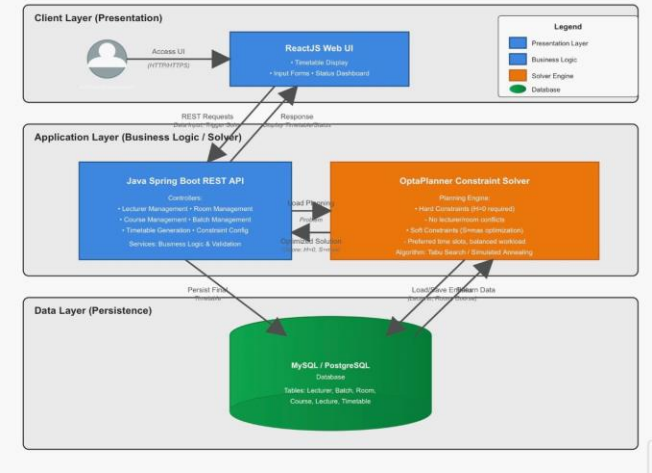


Fig.1 System Architecture

Data integrity and reliable retrieval are paramount. The system leverages a relational database schema optimized for the UCTP entities. Data is extracted from the database, transformed into the Opta Planner planning domain model objects in the Application Layer, and then passed to the solver. The successful timetable solution is persisted back into the database upon completion. The relationships between entities Lecturer, Room, Course, Time Slot, and Batch are complex and foundational to enforcing constraints. This structure is visualized in the Entity-Relationship (ER) Diagram below.

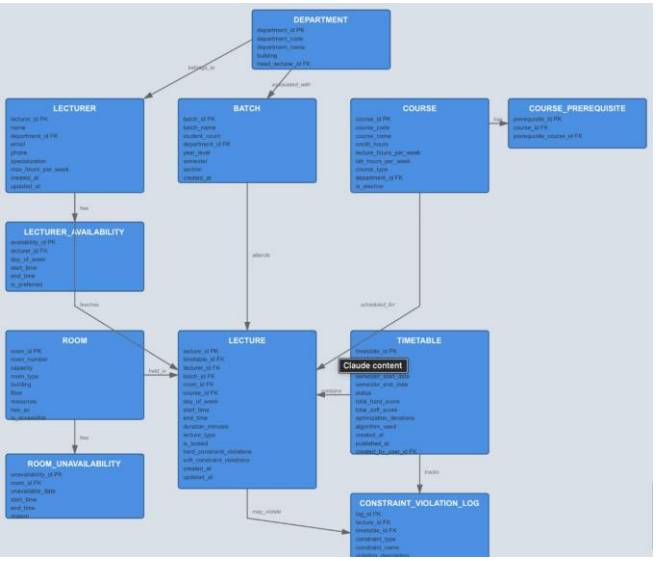


Fig.2 ER Diagram

Entity-Relationship (ER) Diagram detailing the core relationships between planning entities (Courses, Rooms, Time Slots, Lecturers, and Batches) in the CampusSmart database schema.

B. Use Case Modelling

To illustrate the high-level functionalities and user interactions, a Use Case Diagram was developed. This diagram clarifies the boundary of the automated system and defines the primary goals of the key actors, such as the Administrator and the System itself, relating directly to timetable generation and maintenance.

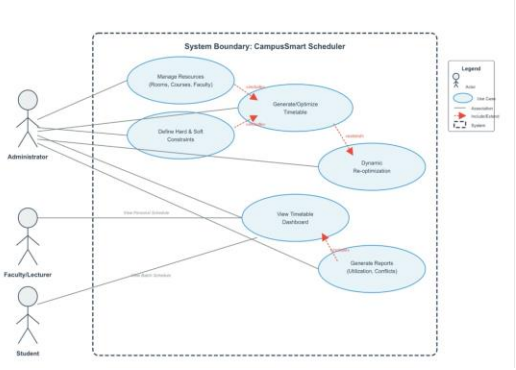


Fig. 3 Use Case Diagram

Use Case Diagram defining the core interactions and goals of system actors, including data input, timetable generation and re-optimisation functions.

IV. MATHEMATICAL FORMULATION

Let L represent the set of lectures, R the set of rooms, and T the set of time slots. The optimisation objective is defined as:

Minimise $F=\sum Hi + \sum WjSj$

where Hi denotes hard constraint violations and Sj represents soft constraint penalties with associated weights Wj.

Room utilisation is calculated as:

U=Allocated seats/Room capacity

Penalties are applied when utilisation falls below a predefined threshold, encouraging efficient use of available space.

V. IMPLEMENTATION DETAILS

The system is implemented using Java and the Spring Boot framework. OptaPlanner is used as the optimisation engine due to its support for constraint modelling and heuristic solving. The frontend is developed using ReactJS, allowing administrators to interact with the system through a web-based interface. A relational database is used to persist institutional data and generated schedules.

A. Technology Stack Selection

The system was implemented using reliable, open-source technologies:

a. Backend: Java Spring Boot, chosen for its stability, performance, and ease of integration with optimisation frameworks.

b. Optimisation Engine: OptaPlanner, providing built-in support for constraint solving and metaheuristic algorithms such as Genetic Algorithms.

c. Frontend: ReactJS, enabling responsive and interactive timetable visualisation.

d. Database: MySQL/PostgreSQL for transactional data persistence.

B. OptaPlanner Integration

Key planning entities include lectures, rooms, and time slots. Score calculation, a performancecritical component, was implemented using the Drools Rule Language (DRL). This approach allows constraints to be expressed declaratively and evaluated efficiently, enabling the system to explore millions of candidate solutions per minute and generate complete timetables within seconds.

VI. TESTING AND RESULTS

The system was tested using a simulated data modelled after a real university environment, including multiple departments, large student batches, 50–100 lecturers, and rooms of varying capacities. The results confirmed that the proposed softconstraint model effectively improves room utilization while maintaining strict feasibility. Full timetable generation was consistently achieved within a few seconds

a. Unit Testing : It examined individual system components in isolation, including constraint validation, optimization routines and database transaction handlers.

b. Integration Testing : it is used to verify seamless communication between the frontend interface, backend services, AI optimization engine and ERP system interfaces.

c. System Testing : It is used to evaluate end to end workflows, tracing the complete process from initial data ingestion throughfinal timetable generation and export functionality.

d. Performance Testing : It tests execution efficiency, optimization accuracy and system responsiveness under varying computational loads.

e. User Acceptance Testing : To gather qualitative feedback from administrative personnel regarding usability, output clarity and practical workflow improvement.

VII. TESTING AND RESULTS

The ReactJS-based user interface plays a crucial role in system usability. Administrators can easily input institutional data, define constraint weights, and trigger optimisation runs. The visualisation module presents the final timetable in intuitive formats for lecturers, students, and administrators, enabling quick verification and adoption.

A. Data Entry and Management Interfaces

The administrative dashboards enable users to set complex constraint parameters, such as lecturer constraints, course priority constraints, and define the values for the weights of soft constraints, for example, the underutilization weight. The data entry of the complex constraints enables the optimisation engine to have an accurate representation of the institution's requirements.

B. Visualization of the Optimized Timetable

The graphical representation component is responsible for generating the optimal solution in a form that is easily interpreted and used by stakeholders such as lecturers and students.

Fig.4 Shows faculty load distribution

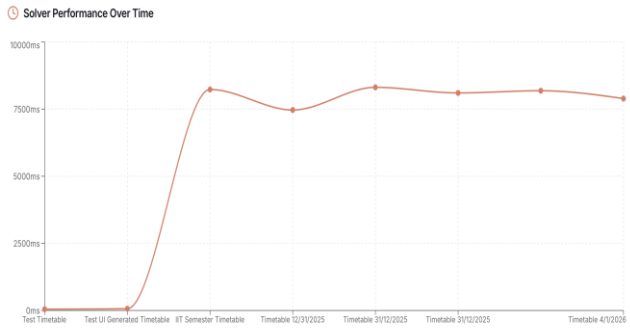


Fig.5 Executio n time analysis of the AI timetable solver demonstrating optimisation stability across different timetable generation

Fig.6 Sample Timetable Screenshot in pdf format

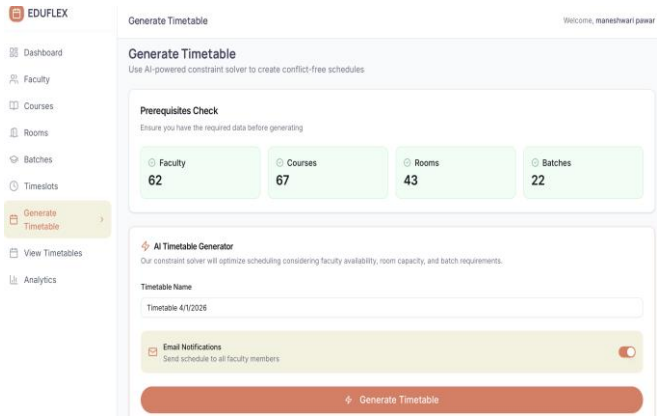


Fig.7 System UI for Timetable Generation illustrating prerequisite check, solver initiation, and notification controls

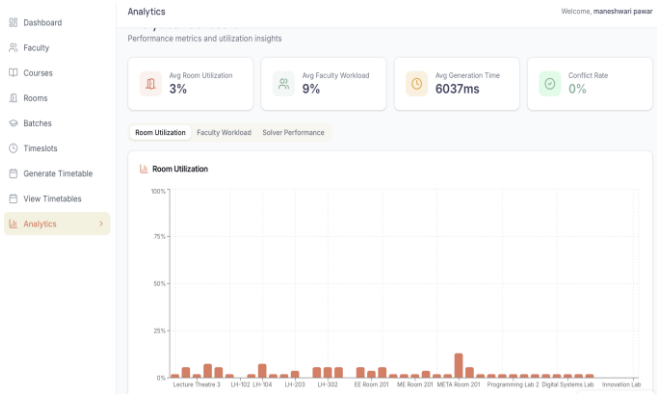


Fig.8 Sample Timetable Screenshot - Room Utilization Dashboard

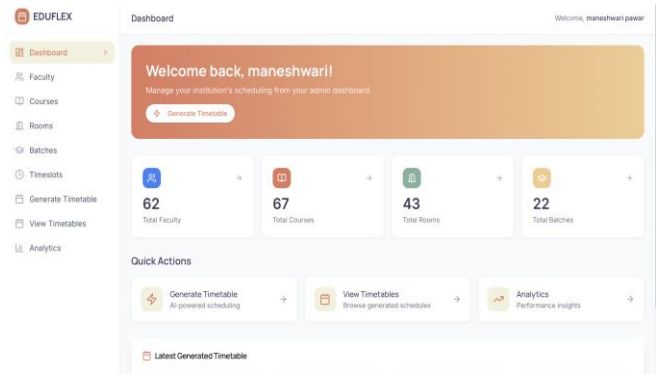


Fig.9 Dashboard view

VIII. CONCLUSION & FUTURE ENHANCEMENTS

This research demonstrates the successful design and implementation of an AI based automated timetabling system focused on smart space utilisation. The CampusSmart Scheduler guarantees conflict-free schedules, significantly improves room utilisation(from below 40% to approximately 90–95%), and reduces timetable generation time from weeks to seconds. These improvements translate into substantial operational and economic benefits for academic institutions. Future work will explore multi-objective optimisation techniques to balance space utilisation with additional factors such as faculty preferences and interbuilding travel time.

Incorporating adaptive learning methods, such as reinforcement learning, to dynamically tune constraint weights based on historical performance is another promising direction.

ACKNOWLEDGEMENT

We also take the opportunity to thank our project guide, Prof. Bhawana Ma'am, for her immeasurable contributions towards the development of our project. It is through her mentorship that we have shaped our approach to the implementation of our project.

We also thank Prof. Sanjay Pal Sir, our Tutor Guardian, for encouraging and supporting us throughout the course of our project. We thank Prof. Shivank Soni Sir, who is handling the Minor Project module, and are thankful to him for the administrative facilitation that helped smooth out our research milestones.

Their joint mentorship and encouragement were crucial to the successful execution of our task.

REFERENCES

[1]. Ahmed, S., Burke, E. K., and Pham, N. (2022). A hybrid optimisation approach for university course timetabling problems. *Journal of Scheduling*, 25(2), 145–160.

[2]. Burke, E. K., Kingston, J. H., and de Werra, D. (2004). Automated timetabling: The state of the art. *European Journal of Operational Research*, 140(2), 266–280.

[3]. Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., and Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1), 177–192.

[4]. Carter, M. W., and Laporte, G. (1998). Recent developments in practical course timetabling. In *Practice and Theory of Automated Timetabling* (pp. 3–19). Springer, Berlin, Heidelberg.

[5]. Gupta, R., and Rao, P. (2020). An evolutionary algorithm-based approach for faculty-oriented university timetable scheduling. *International Journal of Advanced Computer Science and Applications*, 11(6), 312–319.

[6]. Smith, J., Kumar, A., and Verma, R. (2019). Genetic algorithm-based solution for university course timetabling problems. *International Journal of Computer Applications*, 178(7), 25–31.

[7]. OptaPlanner Documentation (2023). Constraint solving and optimisation framework. Red Hat Inc.

[8]. Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2), 87–127.