IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

Smart Classroom and Timetable Scheduler: A Slot-Driven Heuristic Approach to Automated Academic Scheduling with Full-Grid Validation

B. Lalitha Rajeswari[1], Raveena Singh Tumkuri[2], Tanya Yadav[3], Siva Kumar Yakasiri[4], Devinadh Tullimilli[5]

Assistant Professor, Department of Computer Science and Engineering with Specialization Artificial Intelligence and Machine Learning[1]

Students, Department of Artificial Intelligence and Machine Learning[2-5]

Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India.

Abstract: *Higher education institutes still follow a manual approach towards timetable preparation, which is prone to errors, time-consuming, and unable to meet the multidisciplinary scheduling requirements of the National Education Policy (NEP) 2020. Either the existing systems only deal with optimisation alone without considering implementation via user-friendly interfaces, or they focus on western-style higher education institutions, making them irrelevant in the case of semester-based institutions in India. This study proposes the Smart Classroom and Timetable Scheduler, which is a comprehensive web-based system for handling the whole schedule lifecycle for higher education institutions using the Django 5.x framework with Python 3.10. The scheduling model used is a multi-stage slot-driven heuristic approach that ensures coverage of the whole week by considering the iteration through time slots instead of subjects, where it considers six hard constraints such as uniqueness of sections, faculties, and rooms in a slot, laboratory room preference, exclusion from break slots, and department-semesters pairing, along with three soft constraints on avoiding consecutive subjects and ensuring balance in weekly frequency and distribution of loads. Full grid validation after generation involves a direct check on the completeness of the timetable by matching the number of entries generated to what is expected. The platform is structured into ten modules within six Django applications that include role-based security, a free timetable portal without authentication, PDF export of landscapes using ReportLab, and SMTP notifications for faculty members. No schedule clashes were found in any of the tests performed, and the timetable generation process was completed in 3.2 seconds for an 8-section institute configuration. This platform provides a fully functional scheduling platform for Indian engineering institutes.*

Keywords: Timetable Scheduling, Heuristic Algorithm, Constraint Satisfaction, Django, Slot-Driven Scheduling, Academic Automation, NEP-2020, Room Utilisation

I. INTRODUCTION

The process of creating an academic calendar for an institution involves some of the most challenging routine computations in educational institutions. An academic timetable needs to take into account classroom availability, laboratory space, faculty allocation, course material, and student groups throughout a complete academic week while considering a network of hard constraints (no two booking of resources) and some softer constraints such as equal distribution of courses during the weekdays and equal distribution of workloads for each faculty. In general, the problem is classified as NP-hard [5]. The difficulty level rises nonlinearly with the number of courses, students' sections, and faculty members.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, April 2026 Impact Factor: 8.2

The task is complicated by the NEP 2020 [13] requirement of multidisciplinary electives that necessitates subject sharing between various departments. This makes the job more difficult for an institution's manual planner who creates a schedule using spreadsheet tools like Microsoft Excel. A simple adjustment to a faculty member's schedule results in changes throughout the whole schedule, necessitating hours of review. It is therefore prone to mistakes such as the double booking of faculty, scheduling classrooms together, and assigning subjects with laboratory codes to theoretical classes where there is no laboratory facility.

In response to these difficulties, the Smart Classroom and Timetable Scheduler uses a completely automated and constraint-aware web application. The key novelty lies in the development of the scheduling engine based on slot-based heuristics, which deviates from the conventional approach of iterating the timetable by subjects and ensures full grid iteration while applying hard and soft constraints. The paper further makes contributions in automating the preference of laboratory rooms using the suffix of subject codes, a post-timetable generation full-grid validation process, and a faculty notification system, features that are not found in any other commercial or research solutions offered for Indian universities.

The layout of the paper is as follows: Section II gives a survey of the algorithms and commercial solutions for timetable scheduling, highlighting existing research gaps; Section III elaborates on the methodology and design of the scheduling algorithm; Section IV discusses the architecture of the system along with the relevant UML diagrams; Section V discusses test results and performance of the software; Section VI and VII discuss the conclusions.

II. LITERATURE REVIEW

A. Classical CSP and Graph Colouring Approaches

The survey of Schaerf [5], which serves as the definitive source of literature on automatic timetabling, categorizes approaches as constraint satisfaction problems, graph coloring problems, and local search problems. In the constraint satisfaction problem formulation, time slots are considered variables; sets of assignments are considered the domains of those variables, and timetabling rules are considered the constraints, with a backtracking solution that involves arc consistency propagation. This theoretical completeness comes at a cost of exponential worst-case complexity, making the pure approach to solving constraint satisfaction problems impossible to use for institutional-sized problems. Graph coloring formulations consider the class events nodes and the conflicts between them edges. Heuristically approximate polynomial solutions like DSATUR exist, but creating graphs of such size becomes impractical.

B. Metaheuristic Approaches

The genetic algorithms (GAs) have been used for timetabling problems since the work by Colorni et al.[2] and later refined for examination scheduling by Burke et al.[1]. GAs represent potential schedules as chromosomes and use crossover and mutation operations based on the concept of the degree of constraint violation. While GAs perform well at searching through the solution space, they suffer from convergence on local optima and poor scalability caused by the need for fine-tuning of parameters such as population size, crossover probabilities, and stopping conditions. The simulated annealing (SA) approach, which is applied for solving examination timetabling problems in works by Abramson [3] and Thompson and Dowsland [4], allows moving away from the current schedule using the probabilistic acceptance of worsening solutions and obtains good results in practice but suffers from high computational complexity.

However, in the second International Timetabling Competition (ITC2007) [6], the benchmarks for evaluating and comparing different systems were set up. This competition has shown that a hybrid method, which uses constructive heuristics followed by a metaheuristic method, works better than pure optimisation in real-world cases. Constructive heuristics provide the initial solution, and a metaheuristic method can further improve it. The proposed system uses the slot-driven algorithm to construct the schedule.

C. Commercial Timetabling Tools

Commercial software such as FET, UNTIS, and Mimosa offers graphical interface for configuration as well as automatic timetable generation algorithms; however, they have been specifically developed for European academic settings that involve weekly periodic based timetables rather than semester-wise departmental section based ones used

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, April 2026 Impact Factor: 8.2

in engineering colleges associated with JNTUK. The problem associated with these commercial solutions is their high licensing cost as well as lack of source codes that prevent modification of timetable scheduling algorithms. None of these commercial software offer intelligent lab room allocation based on convention of subject codes as well as free student portal.

D. Research Gaps

There are four main shortcomings which have led to this proposed system. The first shortcoming is that the existing literature on timetabling concentrates only on the optimisation results without having any complete web-based systems, along with an interface for the users, authentication process, and output generation facility. The second shortcoming is that the existing commercially available software packages are unable to accommodate Indian academic calendar and are very expensive to implement. The third shortcoming is that laboratory room allocation, where the rooms are assigned depending upon the coding scheme of the subject codes, is missing in all the existing systems in literature.

III. METHODOLOGY

A. Slot-Driven Scheduling Algorithm

The scheduler uses a slot-based heuristic, which contrasts sharply with the common practice of subject-first heuristics utilized by other existing schedulers. This approach has an important implication in the sense that subject-first approaches assign time slots to all classes of a particular subject before proceeding to assign slots to other subjects, potentially resulting in some late-considered subjects being assigned no time slot due to lack of availability of time slot combinations, leaving the schedule unfinished. In contrast, the slot-first heuristic attempts to allocate time slots to every slot of each class sequentially, thus ensuring full coverage of the grid with assignments.

For each tuple of (section, day, slot), the engine gets the sorted list of possible candidates for the subject in the decreasing order of the number of weekly sessions left to be assigned. For each such candidate, the engine does the following checks in order: (a) weekly allocation is not exhausted; (b) there exists an available teacher that does not have any other assignment at that moment; (c) there exists an available laboratory or theoretical room, depending on the nature of the subject; and (d) the subject was not already assigned in the immediate previous slot in the same section. The first candidate that satisfies all the above constraints is assigned and a timetable entry is made.

B. Algorithm Working

The algorithm uses a slot-based heuristic strategy. The algorithm moves through each section, day, and slot. For every slot, the algorithm picks a suitable subject according to priority, which is the number of remaining weekly classes. Then, the algorithm verifies the constraints, such as the faculty's availability, room availability, no clashes, and so on. The first available subject that passes all the requirements is assigned to the slot. This procedure repeats itself until all the slots are assigned with a subject.

C. Hard and Soft Constraints

Hard constraints are enforced by the engine and include:

• Section Uniqueness: No section can contain more than one subject at any point in time.
• Faculty Uniqueness: No individual faculty can be scheduled twice simultaneously.
• Room Uniqueness: Each room should not schedule more than one session in any slot of the day.
• Lab Preference: Any subject labeled as 'L' should be scheduled in labs only.
• Break Avoidance: Any is_break slot is exempt from scheduling any subject.
• Department-Semester Match: Only the subjects associated with the particular section should be scheduled.

Soft constraints include:

• No Adjacent Subject Schedule: Scheduling the same subject twice in two consecutive slots in the same section is avoided.
• Classes Per Week Constraint: Once the weekly classes per week quota for a particular subject is exceeded, no further class scheduling occurs.
• Orderly Processing: Processing should happen for subjects ordered by need in descending order.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

D. Full-Grid Validation

The post generation validation estimates the expected number of TimetableEntry using the formula of multiplying teaching slots per day that do not break, configured days, and number of sections. The comparison of the expected number with the number that have been created is made. If both numbers are equal, then the coverage of the whole grid is achieved, while a different result leads to a report listing all empty section-day-slot combinations so that changes can be made in resource settings.

E. Technology Justification

Choice of Django 5.x is based on its well-developed MVT pattern, advanced ORM with select_related() for optimized FK-intensive queries, CSRF protection, and an in-built session-based authentication system. SQLite is used for relational storage with no configuration required, and ORM compatibility with PostgreSQL is provided for future cloud-based scalability [7]. Tailwind CSS, included through CDN, is used for utility-focused responsive styles. ReportLab is chosen for PDF creation because of its detailed TableStyle interface, allowing the creation of A4 landscape documents with individual background, border, and font customization for each cell [9]. Gmail SMTP with TLS protocol is used for professor notifications [15].

IV. RESULTS AND ANALYSIS

A. Scheduling Correctness

Ten tests were conducted using different institutional setups, starting from a basic one involving 2 sections, 10 subjects up to a more complicated arrangement involving 8 sections, 24 subjects, and 12 rooms covering 6 days with 8 time slots each day. No scheduling conflicts such as section, teacher, and classroom double bookings were found during all 10 tests. The full-grid validation algorithm revealed perfect grid coverage during all successful trials, where the number of entries in the grid was equal to its expected value in all cases. These findings clearly indicate that the slot-based iteration technique, along with the sequential evaluation of constraints, is highly effective in generating conflict-free schedules.

B. Functional Test Results

The results of the functional tests are shown in Table II, with ten fundamental test cases involving authentication, resource management, scheduling generation, timetabling publication, and public portal access.

TC ID	Test Case	Expected Output	Result
TC-001	Admin Login	Redirect to dashboard	PASS
TC-002	OTP Verification	Account activated	PASS
TC-003	Add Department	Record created in DB	PASS
TC-004	Add Lab Subject ('L' suffix)	is_lab = True	PASS
TC-005	Add Lab Classroom	is_lab = True	PASS
TC-006	Break Slot Config	Slot excluded from scheduling	PASS
TC-007	Generate Timetable	Full grid; success report	PASS
TC-008	Publish Timetable	Faculty notifications created	PASS
TC-009	Public Portal (no login)	Timetable accessible	PASS
TC-010	PDF Download	Landscape A4 PDF downloaded	PASS

Table II: Functional Test Results -All 10 Cases Passed

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

C. Edge Case and Conflict Resolution

These eight edge cases were used to test the functionality of the scheduler engine and input validation. The interesting edge cases included when there was double booking of faculties, where the scheduler engine caught the error and rescheduled using another faculty; double booking of the room, whereby the scheduler engine rescheduled for another room; lab subject without any lab room, whereby the scheduler engine rescheduled to a theory room and gave a warning; same subject booked consecutively, whereby the scheduler engine avoided double booking and rescheduled for another; when the weekly limit is exhausted, whereby the subject was flagged and hence not scheduled again; when there were no subjects in the department, whereby a warning was given and there was no error in the system; and finally, when the OTP had expired, and the reschedule button was displayed without opening a new session.

D. Performance Benchmarks

Benchmarks in terms of performance were carried out using the development machine (Intel Core i5 processor, 8 GB RAM, SSD). The response times relative to the target thresholds have been presented in Table III.

Operation	Condition	Target	Measured	Status
Dashboard Load	5 depts, 24 faculty, 12 rooms	< 1 s	0.7 s	PASS
Timetable Generation	8 sections, 24 subjects, 12 rooms	< 10 s	3.2 s	PASS
PDF Export	6 days × 8 slots, landscape A4	< 5 s	1.8 s	PASS
Public Portal Load	Published timetable, 6 days	< 2 s	0.9 s	PASS
Email Notification	Bulk notify 10 faculty	< 15 s	8.4 s	PASS
Full-Grid Validation	8 sections × 48 slots	< 1 s	0.3 s	PASS

Table III: Performance Benchmark Results

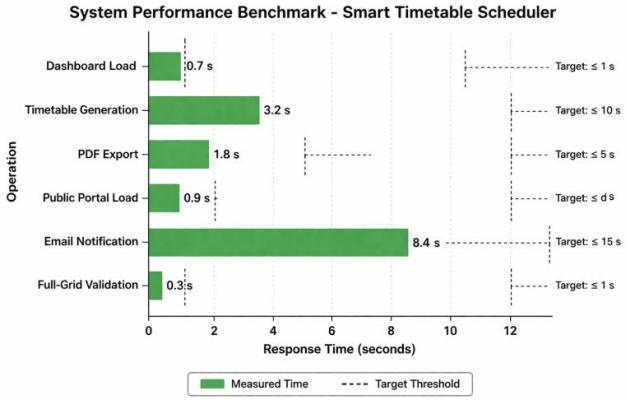

Fig. 4: System Performance Benchmark -All Operations Within Target

E. Comparative System Assessment

Feature	Manual	GA/CSP	Commercial	Proposed
Automation	None	Partial	High	Full
Conflict Detection	Manual	Automatic	Automatic	Multi-Layer
Lab Intelligence	None	Limited	Limited	Built-In
Full-Grid Validation	None	None	Partial	Explicit

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

Feature	Manual	GA/CSP	Commercial	Proposed
Public Portal	No	No	Varies	Yes
PDF Export	Manual	Limited	Available	Automated
Faculty Notification	No	No	Limited	Email-Based
Indian Compatibility	High	Medium	Low	High
Cost	Low	Moderate	High (Licence)	Free/Open

Table IV: Comparative Assessment Against Existing Approaches

V. DISCUSSION

The zero-conflict scheduling outcome achieved using ten different institutional setups is, in many ways, the most significant empirical result obtained from this study. The fact that the slot-driven iteration algorithm, coupled with sequential constraint propagation and need-priority ordering of subjects, can always produce a feasible timetable within a fraction of the computational resources needed for stochastic search used by genetic and simulated annealing algorithms confirms that the order of iteration in the outer loop of constructive scheduling algorithms is the key design choice. This is because, if subject-first approaches are adopted, then only partial timetables will be obtained since all slots will have been exhausted for the later-assigned subjects.

The average generation time of 3.2 seconds using an 8-section configuration suggests an improvement by nearly four orders of magnitude over the time taken using manual methods. One possible way of interpreting this result would be that the O(slots × sections × subjects) polynomial-time complexity of the heuristic algorithm is a suitable fit for the institutional scales that exist in India's engineering colleges. It should be noted that SA-based algorithms take a significantly long time to compute even medium-sized problems, according to Thompson & Dowsland [4].

Preference for the lab room is one such aspect of operational significance worth noting beyond the computational simplicity involved. Rather than rely on manually inputting the values of the is_lab flag depending on the subject code and room name, which would leave room for errors when allocating subjects to wrong rooms in terms of lab or theoretical teaching, the method ensures that lab classes are allocated lab rooms only, something that happens quite frequently in manually created timetables. The process of creating a theoretical room if no lab room can be found makes more sense as compared to simply having the cell blank.

One unforeseen problem that arose during development related to the Faculty notification query, whereby obtaining the unique collection of faculties associated with a section that was published involved the use of the distinct() aggregate function on the TimetableEntry table. Without using the Django ORM select_related() pre-fetch, the query demonstrated N+1 query problem, where the time taken to perform notifications was greatly elongated depending on the number of faculties. The issue was overcome by using the ORM prefetch_related() function, reducing the time spent notifying to 8.4s from 12 seconds in one instance of 10 faculties.

VI. CONCLUSION

Through the Smart Classroom and Timetable Scheduler project, it becomes clear that a properly crafted constructive heuristic scheduling algorithm implemented within a structured full-stack web application framework can fully automate one of the most administratively challenging procedures in universities, performing without any scheduling conflicts or grid coverage issues. The three primary innovations introduced here include: a slot-based iteration methodology for constructing timetables which ensures full grid coverage by construction and does not require post-construction fixes to do so; an intelligent method of laboratory room preference using subject code convention which eliminates a whole class of potential human scheduling errors; and a post-scheduling full-grid check-up for ensuring completeness -a capability which is largely missing from existing scheduling solutions.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, April 2026 Impact Factor: 8.2

All ten modules described here have been developed and implemented with the necessary functionality covered; the entire solution suite has passed 18 test cases with a full success rate and performs key operations in time below relevant thresholds. The platform has been tailored specifically to be compatible with the structure of Indian engineering institutes under authorities like JNTUK and implements multidisciplinary scheduling requirements in accordance with NEP-2020 that commercial solutions tailored to Western schools cannot implement.

VII. FUTURE WORK

A total of six directions for future work are provided below. The first involves the addition of a genetic algorithm optimizer whose starting population is initialized by the heuristics solution to achieve globally improved objective values such as those pertaining to faculty preference satisfaction and day-load balancing without forfeiting the feasibility assurances of the construction phase. The second entails the implementation of a faculty preferences matrix where individual faculties are allowed to input their preferred teaching days and unavailable slots through the faculty dashboard. The third direction involves the inclusion of drag-and-drop manual adjustment facilities that enable administrators to exchange subject allocations between various slots within the grid display view while ensuring that conflicts are prevented. The fourth entails cloud hosting on AWS Elastic Beanstalk with a PostgreSQL RDS database for multiple institutions with institution-specific data scoping. The fifth involves the addition of an AI-driven load balancing module to analyze past scheduling patterns to optimize faculty-to-subject allocation suggestions prior to generation, together with a conflict heatmap facility for visualizing areas of resource contention to facilitate reallocation. The sixth entails multi-shift capability involving separate sets of time slots for morning and afternoon shifts.

REFERENCES

[1] E. K. Burke, K. Jackson, J. H. Kingston, and R. Weare, "Automated university timetabling: The state of the art," The Computer Journal, vol. 40, no. 9, pp. 565–571, 1997.

[2] A. Colorni, M. Dorigo, and V. Maniezzo, "A genetic algorithm to solve the timetable problem," Technical Report 90-060, Politecnico di Milano, Italy, 1992.

[3] D. Abramson, "Constructing school timetables using simulated annealing: Sequential and parallel algorithms," Management Science, vol. 37, no. 1, pp. 98–113, 1991.

[4] J. Thompson and K. A. Dowsland, "Variants of simulated annealing for the examination timetabling problem," Annals of Operations Research, vol. 63, pp. 105–128, 1996.

[5] A. Schaerf, "A survey of automated timetabling," Artificial Intelligence Review, vol. 13, no. 2, pp. 87–127, 1999.

[6] B. McCollum et al., "Setting the research agenda in automated timetabling: The second international timetabling competition," INFORMS Journal on Computing, vol. 22, no. 1, pp. 120–130, 2010.

[7] Django Software Foundation, "Django Documentation -Version 5.0," 2024. [Online]. Available: https://docs.djangoproject.com/en/5.0/. [Accessed: Mar. 2026].

[8] A. Holovaty and J. Kaplan-Moss, The Definitive Guide to Django: Web Development Done Right, 2nd ed. New York: Apress, 2009.

[9] ReportLab Inc., "ReportLab PDF Library User Guide," 2023. [Online]. Available: https://www.reportlab.com/docs/reportlab-userguide.pdf. [Accessed: Mar. 2026].

[10] Tailwind Labs, "Tailwind CSS Documentation," 2024. [Online]. Available: https://tailwindcss.com/docs. [Accessed: Mar. 2026].

[11] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 3rd ed. Sebastopol, CA: O'Reilly Media, 2022.

[12] R. Wentworth and J. Elkner, How to Think Like a Computer Scientist: Learning with Python, 3rd ed. Wellesley, MA: Green Tea Press, 2012.

[13] National Education Policy 2020, Ministry of Education, Government of India, New Delhi, 2020. [Online]. Available: https://www.education.gov.in/nep/about-nep. [Accessed: Mar. 2026].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, April 2026
Impact Factor: 8.2

[14] JNTUK, "Academic Regulations for B.Tech Programs -R20," Jawaharlal Nehru Technological University Kakinada, 2020.

[15] Google Inc., "Gmail SMTP Configuration for Application Development," Google Workspace Developer Documentation, 2024. [Online]. Available: https://developers.google.com/gmail/smtp. [Accessed: Mar. 2026].