International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Schedulify: A Hybrid Approach for Automated University Timetable Generation

Biju Balakrishnan, Abdul Aziz Khatri, Hemang Korane, Yeshang Upadhyay, Eyakramul Hussain, Akshay Nugurwar

Department of Computer Science & Engineering-Industry Embedded Program, Parul Institute of Engineering & Technology, Parul University, Vadodara, Gujarat, India

Abstract- This project provides a hybrid solution for the automated generation of timetables through the combination of Linear Programming (LP) and Constraint Satisfaction Problem (CSP) solutions. It's intended to address the NP-hard problem of university course timetabling, which involves the satisfaction of a considerable number of conflicting constraints and preferences. The method for solving this problem involves two distinct steps. In the first step, a linear programming method is employed to determine the optimal assignment of teachers to subjects. This is a global optimisation technique that aims to maximize the satisfaction of teacher preferences and ensure a well-balanced allocation of teaching loads for all faculty members. In the second phase, a CSP algorithm with backtracking is applied to these optimal assignments to further assign time slots and classrooms. This phase is handled by both hard and soft constraints. Hard constraints, such as the unavailability of a teacher or a classroom, are represented by hard constraints that should not be violated to avoid any scheduling conflicts. Soft constraints, such as teacher time slot preferences and constraints on the lecture interval, are employed to filter the most optimal assignments from the pool of feasible solutions to enhance the quality of the schedule.

Keywords- Automated Timetable Generation, Hybrid Approach/ Hybrid Algorithm, Linear Programming (LP), Constraint Satisfaction Problem (CSP), University Course Timetabling Problem (UCTP), NP-hard/ Combinatorial Optimisation, Hard Constraints and Soft Constraints, Backtracking.

I. INTRODUCTION

Every semester, academic institutions grapple with the well-known NP-hard combinatorial optimisation problem known as the University Course Timetabling Problem (UCTP) (. Under a variety of restrictions, it entails arranging a series of lectures and lab sessions into a small number of time slots and classrooms. These constraints are typically divided into two categories: hard constraints, which must be satisfied for a timetable to be considered valid (e.g., a teacher cannot teach two classes simultaneously), and soft constraints, which are desirable but not strictly mandatory (e.g., a teacher prefers morning classes).

There is a great reason to automate this process. Many schools are still utilising the traditional technique of scheduling, which is incredibly time-consuming and often full of errors. This process often does not create an optimal or even good schedule, which can lead to many conflicts and dissatisfaction among teachers and students. Teachers' morale and teaching quality can be affected if they are assigned schedules that do not consider their preferences regarding work-life balance or teaching subjects that are not their area of expertise (e.g., [8]). While numerous automated approaches exist, there is still a research gap in developing a practical system that effectively balances the global goals of preference maximisation and workload fairness with the local, complex logic of conflict-free slotting. Many existing methods either focus solely on feasibility (finding any valid schedule) or become computationally intractable when trying to optimise for too many variables at once. This paper introduces a novel hybrid approach that decouples the problem into two manageable phases. Our key contributions are A hybrid approach that uses a constraint satisfaction problem for detailed scheduling and linear programming for the best teacher-subject-division assignment.

A formal model that explicitly incorporates and maximises teacher preferences and ensures balanced workloads in the initial assignment phase. A flexible CSP algorithm that

dynamically handles different session types, such as single-slot lectures and consecutive double-slot lab sessions.
A practical implementation that demonstrates the viability of this approach in producing high-quality, conflict-free timetables.

II. RELATED WORK

The UCTP has been extensively studied, leading to a variety of proposed solutions that can be broadly categorised (e.g., [1], [6] and [11]). Many researchers have modelled the UCTP as a pure CSP (e.g., [9]). In this formulation, timetable events are variables, and their domains are the available time slots and rooms. Constraints are then applied to ensure no conflicts occur. Standard backtracking search, often enhanced with heuristics, is used to find a feasible solution. While effective at satisfying hard constraints, pure CSP approaches often struggle to effectively incorporate and optimise for many soft constraints without a clear objective function, which motivates our hybrid design.

Linear Programming is a powerful mathematical discipline for solving optimisation problems involving a linear objective function subject to linear inequality constraints (e.g., [4] and [10]). Since its development by G. B. Dantzig in 1947, its applications have been dominated by planning and scheduling problems. An ILP approach can guarantee a mathematically optimal solution to the assignment problem. However, formulating the entire UCTP—including slot and room assignments—as a single ILP model often results in an enormous number of binary variables, making the problem computationally intractable for all but the smallest institutions. Its primary strength lies in resource allocation problems, which we leverage in our first phase.

Heuristics and Metaheuristics is the largest category of approaches and includes methods like Genetic Algorithms (GAs), Simulated Annealing, and Tabu Search. GAs are particularly adept at exploring large search spaces to find high-quality solutions that satisfy a mix of hard and soft constraints. They are flexible and can handle complex, non-linear objectives. Their main drawback is that they do not guarantee optimality and may not even find a feasible solution in a reasonable amount of time if the problem is too tightly constrained.

Our work is distinct in that it does not rely on a single methodology but instead creates a symbiotic relationship between two. We use LP to do what it does best—global optimisation of assignments—and then use CSP to handle what it does best—finding a valid configuration based on a set of logical constraints.

III. PROBLEM FORMULATION

To formally define the problem, we've identified the core entities and constraints. Entities like Teachers, Subjects, Divisions, Classrooms, Time Slots, representing discrete time intervals (e.g., Monday 9-10 AM).

Hard Constraints (Inviolable):

- **HC1 (Teacher Uniqueness):** A teacher cannot be assigned to more than one class in the same time slot.
- **HC2 (Division Uniqueness):** A division cannot attend more than one class in the same time slot.
- **HC3 (Classroom Uniqueness):** A classroom cannot host more than one class in the same time slot.
- **HC4 (Session Continuity):** Subjects designated as DOUBLE_SLOT (e.g., labs) must be assigned to two consecutive time slots in the same classroom.

Soft Constraints (Desirable):

- **SC1 (Teacher Preference):** Assignments should maximise the preference scores given by teachers for specific subjects and time slots.
- **SC2 (Workload Balance):** The total number of weekly hours assigned to each teacher should be as evenly distributed as possible.
- **SC3 (Subject Spacing):** Avoid scheduling the same subject for the same division multiple times on the same day.

Mathematical Representation:
Phase 1 (LP): Assume a binary decision variable with values of 0 otherwise, and if the teacher is assigned to teach, subject to a division. Let the preference score of the teacher for the subject. The objective is to maximise subject to workload and assignment constraints.
Phase 2 (CSP): Each required teaching session (a teacher-subject-division tuple from the LP output) is a variable. The domain for each variable is the set of all possible (time_slot, classroom) tuples. The hard constraints are modelled as All

Different constraints on the resources (teachers, divisions, classrooms) for each time slot.

IV. METHODOLOGY

Our proposed solution is a two-phase hybrid algorithm. Phase 1: LP for Optimal Teacher-Subject-Division Mapping
The first phase addresses the strategic question of "who teaches what to whom?" It formulates this as a linear programming problem to find the best possible set of assignments before considering the temporal "when" and “where” (e.g. [4] and [12]). This involves maximising a linear objective function subject to a set of linear inequality constraints.

Objective Function:

```
prob+=(
SCORE_WEIGHT* pulp.lpSum(total_score_terms)
+PRIORITY_EPS* pulp.lpSum(priority_terms)
- MAXWORK_PENALTY * max_used
), "Maximize_Total_Satisfaction"
```

Constraints:
Every subject required by a division must be assigned to exactly one qualified teacher. The total hours assigned to each teacher must not exceed their predefined maximum workload. The output of this LP solver is a definitive list of teaching assignments, for example:[(Teacher_A, Subject_CS101,Division_1), (Teacher_B, Subject_EE201, Division_2), ...].

Phase 2: CSP / Backtracking for Slot and Classroom Assignment
The second phase takes the optimal assignments from Phase 1 as input and solves the scheduling puzzle. This is modelled as a Constraint Satisfaction Problem, a common approach for such problems and is solved using a backtracking algorithm (e.g., [14]). The algorithm iterates through each assignment from the LP output and attempts to place it in a valid (time_slot, classroom) pair. If at any point an assignment cannot be placed, the algorithm backtracks to the previous assignment and tries a different placement for it. The search proceeds by first selecting a time slot and then a classroom. For a SINGLE_SLOT session, the algorithm searches for one available slot. For a DOUBLE_SLOT session, it specifically looks for two consecutive available slots. Before placing an assignment, the algorithm checks all hard constraints (HC1-HC3). It verifies that the selected teacher, division, and classroom are all free during the chosen time slot(s). The list of potential time slots for an assignment is sorted based on soft constraints. For example, slots that match a teacher's preferred teaching hours are tried first. This heuristic guides the search towards higher-quality solutions more quickly.

V. IMPLEMENTATION

The system was implemented as a proof of concept using a modern technology stack chosen for its rapid development capabilities and robust libraries (e.g., [2], [3] and [7]). The entire backend logic was implemented in Python 3. The powerful Pandas library was used for data manipulation and for representing availability matrices. The linear programming model in Phase 1 was built and solved using the PuLP library, an open-source LP modeller. The Django ORM was used for data persistence and management. Core entities (teachers, subjects, etc.) and their availability were represented using Pandas DataFrames. For example, teacher availability was stored in a matrix where rows represented teachers and columns represented all available time slots in the week, with Boolean values indicating availability. This structure allowed for efficient lookups during the CSP search. To improve performance and avoid getting stuck in unproductive search paths, several heuristics were implemented (e.g., [5]). Before starting the CSP search, the list of assignments (from the LP output) is randomly shuffled. This helps prevent the algorithm from repeatedly failing on the same difficult-to-place assignment early in the search. The candidate slots for each assignment are also sorted to prioritise those that satisfy soft constraints.

VI. RESULTS AND DISCUSSION

Experimental Setup
The proposed system was evaluated using a dataset modelled closely after a real academic department’s timetable structure, with controlled modifications to test scalability and constraint interactions. The environment consisted of 9 classrooms and subjects distributed across three semesters with differing workloads and departmental diversity:
For Semester 7, 5-day workweek, 5 subjects (all from the same department) with varying numbers of lectures and lab sessions, divided into 2 divisions.

For Semester 5, 6-day workweek, 6 subjects from 4 different departments, distributed across 7 divisions.
For Semester 3, 6-day workweek, 5 subjects from 4 departments, divided into 4 divisions.

A total of 19 teachers were included, covering all 4 departments. The availability for each resource (teacher, classroom, and division) was modelled as a 48-character binary string, representing eight daily time slots over a six-day week. Only teachers relevant to the listed divisions and subjects were considered, ensuring realistic inter-departmental overlap and workload balancing scenarios.
This dataset structure was intentionally chosen to simulate real departmental conditions while also enabling controlled stress testing. By incrementally modifying the classroom-to-division ratio, the system's performance was analysed under varying constraint densities.

Observations and Analysis
The two-stage approach of the system, using Linear Programming (LP) for the mapping of teachers, subjects, and divisions, and then Constraint Satisfaction/Backtracking for slot and classroom assignment, has shown robustness in producing conflict-free timetables under practical departmental settings.

The LP stage was seldom a bottleneck. It always reached an optimal or near-optimal solution in seconds, even for larger input sizes. Only when the problem was infeasible (for instance, when the total teaching requirement exceeded the actual faculty strength) did the LP solver fail to produce a solution. Otherwise, LP performed well in preference balancing and allocation with no noticeable slowdowns.

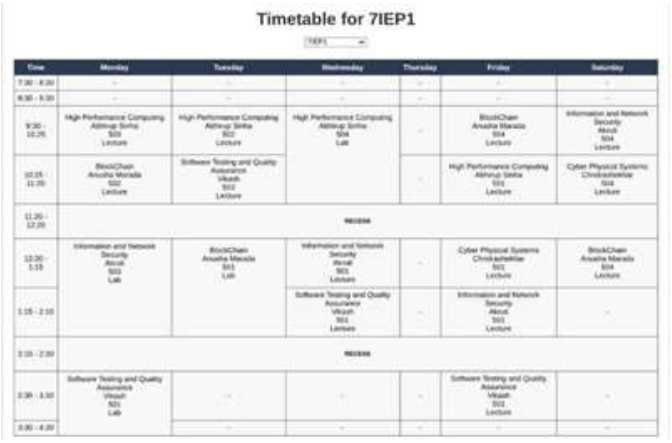
By contrast, the CSP/backtracking phase dominated the computation. The point at which the performance began to degrade was when the ratio of divisions to available classrooms became substantially large. In this case, the algorithm tended to experience infeasible solutions, causing it to "thrash" as it searched for a combination of time slots and classrooms that would satisfy the constraints (e.g., [15]). This points to the fact that the performance bottleneck is not the number of assignments but rather the complexity of the scheduling constraints, particularly the overlapping availability of teachers, classrooms, and time slots. When these constraints become too tight, the search space breaks into smaller, less efficient pieces, causing simple chronological backtracking to be inefficient.

In the given setup (19 teachers, 9 classrooms, and multiple semesters with 3-7 divisions), the system was able to produce a valid timetable after a few trials. This demonstrates that, under the assumption that resource availability is balanced, the suggested hybrid LP-CSP approach is robust and effective for a typical academic scheduling problem.

Performance Metrics
In all successful runs, the algorithm produced a timetable with 0 hard constraint violations, demonstrating its correctness. If no solution was found, it terminated gracefully, indicating an over-constrained problem.

The system's search strategy is designed to satisfy soft constraints whenever feasible. By sorting candidate time slots based on heuristics (such as teacher time preferences), the algorithm systematically prioritises and attempts to place assignments in higher-quality slots first. This ensures that if a placement that satisfies a soft constraint exists without violating any hard constraints, it will be found. The degree of satisfaction is thus a function of the problem's density rather than a fixed percentage.

The standard deviation of assigned weekly hours across all teachers was consistently low (e.g., +/- 2 hours from the mean), indicating that the LP phase was effective at balancing workloads. The system successfully scheduled 100% of the required assignments.



Sample Timetable Output (Division 7IEP1)

Discussion: The results strongly indicate that the hybrid LP+CSP approach is highly effective. The LP phase successfully handles the global optimisation challenge, providing a high-quality set of assignments that already respect preferences and workload balance. This significantly prunes the search space for the subsequent CSP phase, allowing the backtracking algorithm to focus solely on the logistical puzzle of slotting. The CSP phase, in turn, is excellent at enforcing the strict, logical rules required for a valid schedule.

The method's primary success is its ability to produce complete and valid timetables that are also "good" from a human perspective. Its main limitation appeared when dealing with larger-scale problems or extremely dense schedules, where the backtracking algorithm's runtime increased noticeably.

VII. LIMITATIONS & FUTURE WORK

While the proposed system is robust, it has several limitations that offer clear avenues for future research and development. The current model does not include constraints related to classroom equipment (e.g., projectors, computers) or specific room types (e.g., lecture hall vs. seminar room). Future work should involve enriching the data model to include these attributes and incorporating them into the CSP constraint checker.

Although the backtracking algorithm works well, its worst-case time complexity is exponential. For extremely large-scale problems (e.g., an entire university), its performance could degrade. While breaking the problem down by department largely solves the issue, some departments may still be too large for the backtracking search to be computationally efficient. Future work should therefore investigate more advanced CSP heuristics like Minimum Remaining Values (MRV), Least Constraining Value (LCV), or alternative decomposition strategies to ensure scalability even within a single, large academic unit.

VIII. CONCLUSION

This paper successfully demonstrated the design and implementation of a hybrid automated timetable system that synergistically combines linear programming and constraint satisfaction problems. The two-phase approach leverages the strengths of each methodology: LP for globally optimal resource allocation and CSP for precise, conflict-free scheduling. The system effectively automates the complex timetabling workflow, producing feasible and high-quality schedules that respect many teacher preferences and ensure equitable workload distribution. This hybrid method proves to be a practical and powerful solution for academic institutions seeking to improve efficiency, fairness, and overall satisfaction in their scheduling process.

REFERENCES

1. Davison, M., Kheiri, A., & Zografos, K. G. (2024) Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling.
2. Muhammad, S.H, Galadanci, B.S, Mustapha, A and Yahaya, A.S(2016). Design and Implementation of an Android and Web-Based University Timetable Customization System. Bayero Journal of Pure and Applied Sciences, 10(1): 320 - 325
3. Sadhana Paladugu (2022). Design Patterns for Effective Front-End Development in Modern Web Applications. International Journal of Leading Research Publication (IJLRP), Volume 3, Issue 12.
4. Schulze, M. A. (1998). Linear Programming for Optimisation. Perceptive Scientific Instruments, Inc.
5. Sowmya, M. D., Prathyusha, T., Hussain, A. K., Latha, I. S., & Praveena, N. (2021). User Interfacing with Automated Timetable Generator. Drugs and Cell Therapies in Hematology,10(2).
6. Thakur, R. K., Agrawal, N. K., & Kumar, P. (2025). A practical approach to college timetable scheduling. Journal of Scheduling, 28(3), 10 pages.
7. Wang Xiaoshu (2020). Optimized Development of Web Front-End Development Technology. Journal of Physics: Conference Series, Volume 1693. DOI: 10.1088/1742-6596/1693/1/012057.
8. Xue, G., Offodile, O. F., Razavi, R., Kwak, D.-H., & Benitez, J. (2025). Addressing staffing challenges through improved planning: Demand-driven course schedule planning and instructor assignment in higher education. European Journal of Operational Research, 295(2), 11 pages.

9. Abdennadher, S., & Marte, M. (2000). University Course Timetabling Using Constraint Handling Rules. Applied Artificial Intelligence, 14(4), 311-325.
10. Almutairi, A., & Elhedhli, S. (2025). From Integer Programming to Machine Learning: A Technical Review on Solving University Timetabling Problems. Computation, 13(1), 10.
11. Abdullah, S., Turabieh, H., Burke, E. K., & McCollum, B. (2008). A Hybrid Approach for University Course Timetabling. IJCSNS, 8(8), 127-131.
12. Chen, Y. (2022). A Mixed-Integer Linear Programming Model for University Course Timetabling Problems. Wageningen University eDepot.
13. Musa, S. N., et al. (2021). A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. IEEE Access, 9, 10643-10658.
14. Marte, M. (2002). Models and Algorithms for School Timetabling - A Constraint-Programming Approach. LMU Munich Publications.
15. Müller, T. (2005). Constraint-based Timetabling. Purdue University.