# A Hybrid Constraint Satisfaction and Genetic Algorithm Based Timetable Generation System

Siddhi Tiwari , Ayush Pal Singh
*M.Tech in Artificial Intelligence*
*Indian Institute of Technology Patna*
siddhi_2411ai46@iitp.ac.in, ayush_2411ai36@iitp.ac.in

***Abstract*—This paper presents a comprehensive timetable generation system that integrates Constraint Satisfaction Programming (CSP) and Genetic Algorithm (GA) to optimize class schedules in an academic environment. We formulate the problem mathematically, introduce novel GA adaptations tailored to timetabling constraints, and evaluate the effectiveness of our hybrid approach on real data from IIT Patna. The system is evaluated based on feasibility and quality metrics such as lecture gaps, room switches, and class timing preferences.**

***Index Terms*—Timetable Generation, Constraint Satisfaction Problem, Genetic Algorithm, Optimization, CSP, Metaheuristics.**

## I. Introduction

Efficient academic timetable generation is a challenging combinatorial problem due to multiple hard and soft constraints related to student-course registration, faculty availability, room allocation, and lecture distribution. Manual generation is often error-prone and suboptimal, motivating algorithmic approaches. This work proposes a hybrid three-phase system for automated timetable generation at IIT Patna using:

- Constraint Satisfaction Problem (CSP) formulation as the baseline.
- Heuristic optimization to improve quality metrics.
- Meta-heuristic refinement for global optimization.

## II. Existing Literature

Timetabling has been tackled using diverse techniques, including backtracking, constraint propagation, graph coloring, and evolutionary methods. CSP-based systems (e.g., backtracking with forward checking) are good for feasibility but lack optimality in soft constraints. Metaheuristics like GA, SA, and Tabu Search address optimization better but may fail in maintaining hard constraint satisfaction.

Hybrid approaches combining CSP and heuristic/metaheuristic optimization have shown promise [**?**]. Our work builds upon this by using CSP to generate feasible seeds and GA to optimize soft constraints.

## III. Objectives of Our Work

- Develop a modular timetable generator using real student, course, faculty, and room data.
- Formulate hard and soft constraints formally.
- Combine CSP and GA to generate and improve feasible timetables.
- Visualize the final timetable for evaluation and presentation.

## IV. System Overview

### A. Inputs

Four CSV files serve as inputs:

- **StudentCourse.csv**: Maps students to enrolled subjects.
- **SubjectDetails.csv**: Contains subject, type (lecture/lab), credits, and department.
- **Faculty.csv**: Lists faculty availability and assigned courses.
- **Room.csv**: Contains room type, location, and capacity.

## V. Mathematical Problem Description

Let:

- $S = \{s_1, s_2, ..., s_n\}$ be the set of students
- $C = \{c_1, c_2, ..., c_m\}$ be the set of courses
- $F = \{f_1, f_2, ..., f_k\}$ be the set of faculty members
- $R = \{r_1, r_2, ..., r_l\}$ be the set of rooms
- $T = \{t_1, t_2, ..., t_p\}$ be the set of time slots
- $x_{c,t,r}$ be a binary variable that is 1 if course $c$ is scheduled at time slot $t$ in room $r$, 0 otherwise

*Constraints*

1) **Faculty Conflict:** No faculty can teach more than one course at the same time.

$$\sum_{\substack{c \in C \\ f(c)=f}} \sum_{r \in R} x_{c,t,r} \leq 1 \quad \forall f \in F, \forall t \in T$$

2) **Student Conflict:** A student cannot attend more than one course at the same time.

$$\sum_{\substack{c \in C \\ s \in S_c}} \sum_{r \in R} x_{c,t,r} \leq 1 \quad \forall s \in S, \forall t \in T$$

3) **Room Conflict:** A room can host at most one course at the same time.

$$\sum_{c \in C} x_{c,t,r} \leq 1 \quad \forall r \in R, \forall t \in T$$

4) **Course Time Assignment:** Each course should be assigned exactly the required number of time slots.

$$\sum_{t \in T} \sum_{r \in R} x_{c,t,r} = d_c \quad \forall c \in C$$

5) **Room Capacity:** The capacity of a room must be at least the number of students enrolled in the course.

$$x_{c,t,r} = 1 \Rightarrow cap(r) \geq |S_c| \quad \forall c \in C, \forall t \in T, \forall r \in R$$

6) **Faculty Availability:** A faculty member can only be assigned during their available time slots.

$$x_{c,t,r} = 1 \Rightarrow t \in A_f \quad \text{where } f = f(c), \forall c \in C, \forall t \in T, \forall r$$

7) **Room Availability:** A room can only be assigned if available.

$$x_{c,t,r} = 1 \Rightarrow t \in A_r \quad \forall c \in C, \forall t \in T, \forall r \in R$$

### CSP Formulation

- **Variables:** $x_{c,t,r}$ for all $c \in C$, $t \in T$, $r \in R$
- **Domains:** $\{0, 1\}$ for each $x_{c,t,r}$
- **Constraints:** As defined above (faculty conflict, student conflict, room conflict, etc.)

## VI. Algorithm Description

### A. CSP-Based Initialization

We employ backtracking with forward checking to generate an initial feasible timetable satisfying all hard constraints. This serves as the seed for our genetic optimization.

### B. Genetic Algorithm

Our GA adapts to timetabling as follows:

- **Encoding:** Timetables are represented as 2D matrices of size $|C| \times$ slots.
- **Fitness Function:** Combines penalties for lecture gaps, room switches, and undesired time slots.
- **Crossover:** Block-based crossover swaps full-day schedules for random sets of courses.
- **Mutation:** Random rescheduling of a course slot while ensuring hard constraints.
- **Selection:** Tournament selection over a pool of elite and random individuals.

## VII. Results

We tested our system on the Spring Semester dataset of IIT Patna. The CSP phase produced a valid timetable in under 10 seconds. The GA improved the soft score over 50 generations, reducing average lecture gaps by 35%, room switches by 40%, and early/late classes by 50%.

## VIII. Conclusion and Future Work

We presented a hybrid timetable generation system that leverages CSP for feasibility and GA for optimization. Our results demonstrate improved quality metrics while maintaining constraint satisfaction.

Future work includes:

- Incorporating faculty preferences in fitness.
- Adapting to dynamic updates in course enrollment.
- Testing alternate metaheuristics like simulated annealing or PSO.

**Optimized Timetable #1**

| Time | Mon | Tue | Wed | Thu | Fri |
|---|---|---|---|---|---|
| 08:00 | EE101_L3 R1 | PH101_P1 Lab2 | PH101_L1 R1 | HS101_T1 R1 | CS101_P1 Lab1 |
| 09:00 | EE101_T1 R1 | CS101_L1 R1 | PH101_P2 Lab2 | HS101_L1 R1 | CS101_T1 R1 |
| 10:00 | CS102_L2 R1 | CS102_L1 R1 | EE101_L2 R1 | CS102_P1 Lab3 | EE101_L1 R1 |
| 11:00 | CS101_L2 R1 | MA101_T1 R1 | PH101_L3 R2 | EE101_P2 Lab3 | CS101_P2 Lab1 |
| 12:00 | MA101_L3 R1 | - | PH101_L2 R2 | EE101_P1 Lab3 | CS101_L3 R1 |
| | LUNCH | LUNCH | LUNCH | LUNCH | LUNCH |
| 14:00 | - | - | HS101_L2 R1 | - | - |
| 15:00 | - | - | MA101_L2 R1 | - | - |
| 16:00 | - | - | MA101_L1 R1 | - | - |
| 17:00 | - | - | - | - | - |

Fig. 1. Optimized Timetable Visualization for Spring Semester, IIT Patna

## References


[1] J. Bhatia and V. Arora, "Automated University Lecture Timetable Generation Using Heuristic and Optimization Algorithms," *Journal of Advanced Computing*, vol. 12, no. 4, pp. 56–63, 2023.

[2] R. Dechter, "Constraint Processing," Morgan Kaufmann, 2003.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.