IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SMART AI POWERED TIMETABLE GENERATOR

[1]SRI SOWNDHARYA.G, [2]SIVARANJANI.S, [3]SRINIDHI.S, [4]SUBASHRI.D

[1]ASSISTANT PROFESSOR, [2]STUDENT, [3]STUDENT,[4]STUDENT

[1,2,3,4]Department of Computer Science and Engineering

Bharath Institute of Higher Education and Research, Chennai, India

Abstract: This project presents an AI-powered academic scheduling system designed to automate timetable generation in educational institutions. Using Genetic Algorithms combined with constraint satisfaction techniques, the system ensures clash-free timetables by preventing teacher and room conflicts while balancing workloads. A full-stack architecture integrates a React.js frontend, Node.js + Express backend, and MongoDB database for persistent storage and multi-user support. Administrators can input student, faculty, and classroom data, and the system dynamically generates optimized timetables with options for visualization and export. Designed for scalability and adaptability, the solution reduces manual effort, improves accuracy, and supports future enhancements such as cloud deployment, calendar synchronization, and workload fairness.

Index Terms **-timetable generation, genetic algorithm, constraint satisfaction problem (CSP), optimization, clash-free scheduling…**

I. INTRODUCTION

Academic scheduling is a fundamental administrative task in schools, colleges, and universities. It involves the allocation of courses, classrooms, and faculty across multiple student groups while adhering to institutional rules and constraints. Traditionally, timetables are prepared manually, a process that is not only time-consuming but also prone to human error. Common issues include overlapping classes, double-booked rooms, misallocated subjects, and uneven distribution of faculty workloads. As institutions grow in size and complexity, manual scheduling becomes increasingly inefficient and difficult to adapt to sudden changes such as faculty absences, new course additions, or room unavailability.

To overcome these challenges, automated scheduling systems have emerged as a promising solution. By leveraging artificial intelligence and optimization algorithms, these systems can generate conflict-free timetables that respect institutional policies while minimizing administrative effort. Among various approaches, Genetic Algorithms (GA) have proven effective for solving complex optimization problems like timetabling. GA simulates the process of natural selection, evolving candidate solutions through operations such as selection, crossover, and mutation. Combined with Constraint Satisfaction Problem (CSP) solvers, Integer Linear Programming (ILP), and Graph Coloring techniques, GA ensures that generated timetables are both valid and optimized.

The proposed system, TimeGene, integrates these algorithms within a modern full-stack architecture. A React.js frontend provides an intuitive interface for administrators to input data and visualize schedules, while a Node.js + Express backend manages logic, APIs, and communication with a MongoDB database for persistent storage. This design enables scalability, multi-user support, and real-world deployment. The system outputs both student and staff timetables, with options for export in formats such as Excel or PDF, ensuring usability across diverse institutional contexts.

By automating the scheduling process, TimeGene reduces manual workload, improves accuracy, and enhances adaptability. It represents a step toward intelligent academic management, offering institutions a scalable, efficient, and user-friendly solution.

Future enhancements include cloud deployment, calendar synchronization, and human-centric features such as workload fairness and comfort optimization, further strengthening its role as a comprehensive academic scheduling tool.

II. PROPOSED METHODOLOGY

2.1 Input Modules

Input modules form the foundation of the scheduling process. They allow administrators to systematically enter institutional data:

2.1.1 Course Data: Includes subject names, credits, required sessions, and elective options. Faculty Data: Captures availability, workload preferences, departmental assignments, and teaching constraints.

2.1.2 Classroom Data: Records room capacities, facilities (e.g., labs, projectors), and availability.

2.1.3 Student Data: Stores enrollment details, electives, and group allocations.

Constraints such as maximum teaching hours, room capacities, and institutional rules are encoded during input. Validation mechanisms prevent inconsistencies, ensuring clean data for optimization. By structuring inputs carefully, the system minimizes errors and enhances scheduling accuracy.

2.2 Constraint Definitions

Constraints are central to timetabling. They are divided into:

2.2.1 Hard Constraints: Must be satisfied. Examples include avoiding overlapping classes, ensuring faculty are not double-booked, and respecting room capacities. Violating these constraints renders a timetable invalid.

2.2.2 Soft Constraints: Preferred but not mandatory. Examples include minimizing consecutive sessions, balancing workloads, and avoiding late evening classes. These constraints improve usability and fairness but can be relaxed if necessary.

The fitness function integrates both types, penalizing violations and rewarding compliance. This ensures that generated timetables are feasible while also optimized for comfort and fairness.

2.3 Optimization Engine

The optimization engine combines multiple algorithms:

2.3.1 Genetic Algorithm (GA): Evolves candidate timetables through selection, crossover, and mutation, gradually improving fitness scores.

2.3.2 Constraint Satisfaction Problem (CSP): Ensures institutional rules are respected, validating candidate solutions.

2.3.3 Integer Linear Programming (ILP): Provides mathematical precision in resource allocation, refining classroom and faculty distribution.

2.3.4 Graph Coloring: Treats subjects as nodes and conflicts as edges, ensuring non-overlapping schedules. This hybrid approach balances adaptability, scalability, and accuracy. GA generates diverse solutions, CSP enforces rules, ILP fine-tunes allocations, and graph coloring eliminates overlaps. Together, they deliver timetables that are conflict-free, efficient, and fair.

III. DATA PREPROCESSING

Data preprocessing plays a critical role in ensuring that the input data is accurate, consistent, and suitable for optimization in the proposed academic scheduling system. The system processes multiple datasets, including course details, faculty information, classroom specifications, student enrollment data, and institutional constraints.

Initially, raw data is collected from institutional records and structured into predefined formats. Data cleaning is performed to remove duplicate entries, handle missing values, and standardize formats such as course codes and time slots. This step ensures data consistency and eliminates anomalies that may affect scheduling accuracy.

Subsequently, data transformation is carried out to convert categorical attributes into numerical representations. Techniques such as label encoding are applied to convert course names, faculty identifiers, and classroom IDs into machinereadable formats. Additionally, faculty availability and classroom occupancy are represented using structured matrices to facilitate efficient processing.

Constraint encoding is a key component of preprocessing. The system distinguishes between hard constraints and soft constraints. Hard constraints include conditions such as avoiding faculty overlap, preventing classroom conflicts, and ensuring room capacity limits are not exceeded. Soft constraints focus on improving timetable quality by minimizing idle periods, balancing faculty workloads, and avoiding consecutive sessions.

The processed data is then structured into a chromosome representation suitable for optimization. Each chromosome encodes a potential timetable solution, where genes represent course assignments mapped to specific faculty, classrooms, and time slots. Finally, validation mechanisms are applied to ensure that the dataset is logically consistent and adheres to institutional requirements before being passed to the optimization engine.

IV. MODEL TRAINING

Unlike traditional machine learning models, the proposed system employs a hybrid optimization approach for timetable generation. The training process is formulated as an iterative optimization problem, where candidate solutions are refined over multiple generations to achieve optimal scheduling.

A. Genetic Algorithm Initialization

The optimization process begins with the initialization of a population of candidate solutions. Each candidate, referred to as a chromosome, represents a complete timetable configuration. These chromosomes are generated randomly while ensuring basic feasibility.

B. Fitness Function Design

A fitness function is defined to evaluate the quality of each timetable. The function penalizes violations of hard constraints such as faculty conflicts and classroom overlaps, while also incorporating soft constraint considerations like workload balance. The objective is to maximize the fitness score by minimizing constraint violations.

C. Selection Mechanism

The selection process identifies high-quality candidate solutions based on their fitness scores. Techniques such as elitism or tournament selection are used to retain the best-performing chromosomes for the next generation, ensuring continuous improvement.

D. Crossover Operation

Crossover is applied to combine features of two parent chromosomes to generate offspring. This operation promotes diversity in the solution space and enables the algorithm to explore new scheduling configurations.

E. Mutation Strategy

Mutation introduces controlled randomness by altering specific genes within a chromosome. This prevents premature convergence and ensures that the algorithm explores a broader search space, improving the likelihood of finding optimal solutions.

F. Iterative Optimization Process

The algorithm iteratively performs selection, crossover, and mutation over multiple generations. With each iteration, the population evolves toward better solutions, reducing conflicts and improving overall timetable quality.

G. Integration with Constraint Satisfaction and Optimization Techniques To enhance performance, the Genetic Algorithm is integrated with additional optimization techniques:

• Constraint Satisfaction Problem (CSP): Ensures strict adherence to hard constraints by filtering infeasible solutions during the optimization process.

• Integer Linear Programming (ILP): Refines feasible solutions by optimizing resource allocation, such as classroom utilization and faculty workload distribution.

• Graph Coloring: Models scheduling conflicts as graph relationships, ensuring that no overlapping sessions occur by assigning distinct time slots.

H. Final Solution Selection

After completing the optimization process, the best-performing chromosome with the highest fitness score is selected as the final timetable. This solution is guaranteed to be conflict-free and optimized according to institutional requirements.

V. RESULT AND DISCUSSIONS

This chapter provides a balanced evaluation of TimeGene, emphasizing both its strengths and limitations while situating it within broader institutional contexts. The system's strengths include conflict elimination, efficiency, adaptability, fairness, usability, and reliability, all achieved through its hybrid optimization framework that integrates Genetic Algorithms, CSP, ILP, and graph coloring. These features ensure that timetables are consistently valid, resource-efficient, and user-friendly. At the same time, the chapter acknowledges limitations such as parameter sensitivity in GA tuning, computational demand for large datasets, complexity in constraint encoding, and the restricted scope to timetabling alone.

By recognizing these challenges, the project demonstrates transparency and provides direction for future enhancements. Comparative evaluation shows that TimeGene outperforms manual scheduling and many existing models by reducing administrative overhead, improving accuracy, and offering scalability. Finally, the institutional implications highlight its transformative role: enhancing administrative efficiency, improving faculty satisfaction, enriching the student experience, supporting curriculum flexibility, and contributing to digital transformation through modern technologies. Overall, TimeGene emerges as a system that not

IV. CONCLUSION

The TimeGene project was conceived to address the persistent challenges of academic timetabling, a problem characterized by complexity, multiple constraints, and dynamic institutional requirements. By integrating Genetic Algorithms (GA), Constraint Satisfaction Problems (CSP), Integer Linear Programming (ILP), and Graph Coloring, the system successfully demonstrated its ability to generate conflict-free timetables, balance faculty workloads, and optimize classroom utilization.

The project's methodology emphasized modularity and scalability. The adoption of a full-stack architecture—React.js for the frontend, Node.js with Express for the backend, and MongoDB for the database—ensured that the system was technically robust and user-friendly. This design allowed administrators, faculty, and students to interact seamlessly with the system, reinforcing the importance of usability in academic scheduling systems as highlighted by Carter and Laporte (1996).

Results validated the effectiveness of the hybrid optimization engine. Timetables generated by TimeGene consistently satisfied hard constraints, while soft constraints were optimized to enhance fairness and usability. Comparative analysis demonstrated clear advantages over manual methods and existing models, echoing Burke et al. (2004), who found that hybrid approaches achieve higher success rates in constraint satisfaction.

The project also emphasized adaptability. The update and adaptation module allowed timetables to be regenerated in real time when constraints changed, reflecting Schaerf's (1999) call for flexibility in timetabling systems. This adaptability ensured that the system remained reliable under dynamic institutional conditions.

In conclusion, TimeGene achieved its objectives, delivering a practical and intelligent solution to academic scheduling. The project advanced the field by demonstrating how hybrid algorithms, integrated within a modern software architecture, can address long-standing challenges in timetabling. Its contributions span technical innovation, methodological rigor, and institutional impact, positioning it as a transformative solution for academic scheduling.

REFERENCES

[1] Romaguera, D., Plender-Nabas, J., Matias, J., & Austero, L. (2024). Development of a web-based course timetabling system based on an enhanced genetic algorithm. Procedia Computer Science, 234, 1714–1721.

[2]Abdipoor, S., Yaakob, R., Goh, S. L., & Abdullah, S. (2023). Meta-heuristic approaches for the University Course Timetabling Problem. Intelligent Systems with Applications, 19, 200253.

[3]Grigoriță, D.-I., Duțuc, P., Cotiugă, I., Boboc, R.- M., & Iftene, A. (2025). Using Aspect Oriented Programming and Monitor Oriented Programming in timetable generation system. Procedia Computer Science, 270, 2562–2571.

[4]Jiang, Y., & Luo, X. (2025). Optimization of university timetables considering students' thermal sensation in classrooms. Energy and Built Environment, 6(3), 834–846.

[5]Siew, E. S. K., Sze, S. N., Goh, S. L., Kendall, G., Sabar, N. R., & Abdullah, S. (2024). A survey of solution methodologies for exam timetabling problems. IEEE Access, 12, 41479–41495.