Adaptive Scheduler: AI Optimization of Academic Timetable

[1]Suraj Kumar Saw, [2]Steffie Lawrence, [3]Sudesh Karunakara, [4]Natra Hrudhika Komal, [5]VijayaKumari G

[1-4]Department of Computer Science & Engineering, SOET, CMR University, Bangalore, India
[5]Assistant Professor, Department of Computer Science & Engineering, SOET, CMR University, Bangalore, India
[1]suraj.kumar@cmr.edu.in, [2]steffie.l@cmr.edu.in, [3]sudesh.k@cmr.edu.in, [4]natra.hrudhika@cmr.edu.in, [5]vijayakumari.g@cmr.edu.in

Abstract **- This paper presents the design and implementation of an AI-Optimized Timetable Generator for academic scheduling, aimed at improving the efficiency and effectiveness of scheduling faculty, subjects, and classrooms in educational institutions. The system leverages a combination of genetic algorithms and machine learning techniques to generate optimal timetables based on multiple constraints, including faculty availability, subject hours, and subject preferences. The web application provides a user-friendly interface where administrators can add faculty and subject data, while the system automatically generates the timetable and provides the option to download it in PDF (Portable Document Format) format. Additionally, feedback from faculty members on the schedule can be collected and examined to continuously improve the timetable generation process. The solution aims to reduce the time and effort required for manual scheduling, thus enhancing the administrative workflow in educational institutions.**

Index Terms **– Artificial Intelligence, Timetable Generation, Genetic Algorithm, Machine Learning, Educational Scheduling, Optimization, Web Application, Constraint Handling.**

I. INTRODUCTION

The increasing complexity of academic scheduling in educational institutions presents a significant challenge. Timetables must be optimized to ensure effective allocation of resources, including faculty, classrooms, and time slots, while also adhering to constraints such as availability, faculty preferences, and subject requirements. Traditional methods of timetable scheduling often result in suboptimal solutions due to the manual effort involved, which is time-consuming and prone to human error. This paper presents an AI-driven approach to solving the academic timetable scheduling problem, titled "Adaptive Scheduler - AI Optimization of Academic Timetable." This system leverages artificial intelligence techniques to automate and optimize the timetable creation process, providing an efficient and effective solution for academic institutions.

The scheduling of academic timetables is a complex problem that involves the coordination of multiple variables and constraints. Educational institutions typically rely on manual or semi-automated systems to create timetables, which may not always result in an optimal solution. Over the years, various algorithms, such as genetic algorithms, simulated annealing, and constraint satisfaction, have been explored to automate and improve timetable generation. However, these solutions often lack adaptability to dynamic constraints and do not fully consider the preferences of stakeholders like faculty and students. With advancements in AI (Artificial Intelligence), particularly machine learning and optimization algorithms, there is a growing opportunity to enhance academic scheduling by creating more adaptive and efficient solutions.

The motivation behind this research is to develop an adaptive AI-based system that can generate optimal academic timetables by considering various dynamic constraints. Traditional methods often fall short in terms of flexibility, efficiency, and accuracy when dealing with real-time changes or multiple constraints. The proposed system uses machine learning models and optimization techniques to automatically adjust and improve the timetable, minimizing conflicts and ensuring a more balanced distribution of resources. The significance of this work lies in its potential to revolutionize the timetable generation process. By automating the scheduling process, institutions can save significant time and resources, allowing administrators to focus on other essential aspects of academic management. Additionally, an optimized timetable leads to better utilization of resources, reduces faculty and student workload, and enhances overall academic productivity.

Academic institutions face challenges in creating timetables that meet all constraints, such as faculty availability, classroom capacity, and subject-specific needs. Balancing these factors with individual preferences often results in conflicts. This project aims to develop an adaptive AI system that automates the timetable creation process while accommodating dynamic changes like faculty unavailability or last-minute room shifts. The system will ensure fairness and efficiency in resource allocation, minimize human error, and reduce the time spent on manual adjustments. Additionally, it will provide data-driven insights and recommendations to administrators based on historical scheduling patterns. Ultimately, the goal is to deliver an optimal, adaptive, and scalable solution for academic scheduling.

The primary objective of this project is to develop an AI-driven academic timetable scheduling system that utilizes machine learning and optimization techniques to automate and enhance the timetable generation process. It seeks to overcome challenges commonly found in traditional scheduling methods, such as inefficiencies, conflicts, and human errors. The system is designed to intelligently allocate faculty, classrooms, and time slots by considering dynamic constraints like availability, preferences, and subject requirements. An adaptive mechanism will allow the timetable to adjust in real-time to accommodate unforeseen changes, such as faculty absences or room issues. The project also focuses on ensuring fairness by balancing faculty workloads while respecting their preferences and institutional policies. Additionally, it aims to provide a scalable solution suitable for institutions of various sizes and complexities, and to evaluate its effectiveness through metrics such as time savings, accuracy, and stakeholder satisfaction. Ultimately, this project aspires to deliver a robust and adaptable system for efficient academic scheduling.

II. LITERATURE REVIEW

The problem of academic timetable generation has been extensively studied due to its combinatorial complexity and the constraints associated with scheduling. Traditional methods often fall short when it comes to accommodating dynamic institutional requirements, soft constraints, and optimal resource utilization. This has led to a growing interest in artificial intelligence and metaheuristic approaches such as genetic algorithms for timetable optimization.

Ghasemi et al. [1] explored the integration of Artificial Bee Colony (ABC) with genetic grouping techniques to solve university course timetabling problems. Their approach highlighted the effectiveness of hybrid algorithms in managing soft constraints and complex datasets like the Socha dataset. Yang and Jat [2] proposed enhancements to genetic algorithms by incorporating guided and local search strategies. Their work demonstrated significant improvements in convergence speed and constraint handling when applied to the university course timetabling problem (UCTP). Siledar and Musande [3] implemented AI-based strategies for smart timetable generation. Their research focused on real-time conflict detection and resource allocation, contributing to more adaptive scheduling systems. A novel adaptive genetic algorithm was developed by Saptarini et al. [4] for high school timetabling. This method dynamically adjusted crossover and mutation rates to improve solution diversity and robustness, particularly in large-scale problems.

Premasiri [5] investigated the application of genetic algorithms to timetable generation at Rajarata University. Their study emphasized practical constraints such as faculty availability, room scheduling, and subject load balancing. An early yet impactful contribution by Srinivasan et al. [6] involved multiple context reasoning to automate timetable generation. Though computationally intensive, their approach laid foundational work for constraint-aware scheduling logic. Recently, Subang et al. [7] proposed a dynamic genetic algorithm-based method to optimize university course scheduling. Their model adaptively adjusted to changes in faculty assignments and subject availability, enhancing both flexibility and efficiency.

These existing works as depicted in Table 1 establishes a strong foundation for leveraging evolutionary algorithms and AI for academic scheduling. Our project, adaptive scheduler, builds on this research by applying a modular, scalable genetic algorithm framework using a Flask-Python backend, enabling real-time feedback integration and enhanced adaptability tailored for academic institutions.

Table 1: Summary of Selected Studies on Timetable Optimization

Study	Method	Application	Results/Findings
Ghasem et al.	Artificial Bee Colony (ABC)	University courses	Effective management of soft constraints; suitable for complex datasets
Wang and Zhat	Enhanced Genetic Algorithm (GA)	University courses	Improved convergence speed and enhanced constraint handling
Siledar et al.	AI – Based Algorithm (GA)	High School	Supports adaptive scheduling and robust detection
Baptarini et al.	Adaptive Genetic Algorithm (GA)	University courses	Improved solution diversity and robustness
Subang et al.	Dynamic Genetic Algorithm (GA)	University courses	Increased flexibility and efficiency

III. METHODOLOGY

The methodology adopted in this project follows a structured pipeline to design, implement, and deploy an AI-based academic timetable scheduler. Each stage of development in the Fig. 1 and Fig. 2 was aligned with the goal of producing optimal schedules that meet institutional constraints while ensuring scalability and usability. The system utilizes genetic algorithms as its core optimization engine, supported by well-structured data flow, rule enforcement, and dynamic front-end visualization. Genetic algorithms have been widely used in optimization problems due to their robustness in solving complex scheduling issues [1], [4], [7]. The following subsections detail each phase of the methodology:

A. Dataset Collection and Preprocessing

The initial phase involved collecting course-related data, including subject names, faculty information, time slots, and days. This data was stored in structured JSON (JavaScript Object Notation) files and represented programmatically as Python lists and dictionaries for easier manipulation. Preprocessing ensured normalization, duplication removal, and consistency across attributes, allowing seamless integration into the scheduling engine. Preprocessing is critical in any machine learning or optimization system to ensure high-quality input data, which significantly affects the performance of the algorithm.

B. Constraint Identification

To generate realistic and acceptable timetables, it was essential to identify both hard and soft constraints. Hard constraints included non-overlapping sessions for faculty, avoidance of duplicate time-slot assignments, and respecting the availability of classrooms and instructors. Soft constraints aimed to improve schedule quality, such as minimizing gaps in faculty schedules or maximizing subject diversity per day. Constraint handling is a key aspect of scheduling algorithms, as it ensures practical solutions that meet real-world requirements.

C. Algorithm Selection and development

Given the NP-hard nature of timetable scheduling problems, a Genetic Algorithm (GA) as shown in Fig. 2 was selected for its efficiency in exploring large search spaces and producing near-optimal results [1], [4]. The GA was developed using Python and designed to encode timetable entries as chromosomes.

Custom operators for selection, crossover, and mutation were implemented to ensure diversity and evolution in the population. Careful tuning of mutation rates and population size was performed to balance convergence speed and solution quality [8].

D. Fitness Function Design

The fitness function was carefully designed to evaluate how well a given timetable met the defined constraints. It rewarded solutions that avoided clashes, respected faculty availability, and maintained diversity in subjects across the week. Penalties were assigned for violations such as duplicate classes or missing slots. The fitness score thus served as a guiding metric for selecting the best candidates in each generation, ensuring that only improved timetables were propagated forward.

E. Generation and Evaluation Loop

The evolutionary loop included initializing a random population, evaluating each individual using the fitness function, and applying selection, crossover, and mutation to generate new candidates. This process was repeated over several generations until a threshold fitness level was achieved or a maximum number of iterations was reached. Each iteration brought the system closer to an optimized timetable configuration, with logs and metrics recorded for analysis [10], [11].

F. Model Selection and Architecture

The overall system architecture combined a Flask-based backend with an HTML (Hyper Text Markup Language) /JavaScript front-end. The optimization logic was housed in Python modules, invoked through API (Application Programing Interface) endpoints served by Flask. The modular structure allowed decoupling of the scheduling logic from the UI (User Interface) and made the system easier to maintain and expand. This separation of concerns also facilitated independent testing and debugging of the backend components [12], [13].

G. API Design and Integration

To ensure smooth interaction between the optimization engine and the user interface, RESTful (Representational State Transfer) APIs were developed using Flask. Endpoints such as /timetable/generate were designed to accept requests, invoke the scheduler, and return generated timetables in JSON format. The API also supported export functionalities. Tools like Postman were used extensively during development to test API behavior, validate response formats, and ensure robustness in error handling.

H. Front End Data Rendering Strategy

The final stage of the pipeline involved designing a simple, user-friendly web interface using HTML, CSS (Cascading Style Sheet), and JavaScript. Upon clicking the "Generate Timetable" button, a fetch request is sent to the backend API. The returned JSON data is dynamically parsed and rendered into a table using JavaScript DOM (Document Object Model) manipulation. An additional button enables downloading the timetable as a PDF using a backend route. This interaction design ensures a seamless experience for end-users, hiding the complexity of the optimization behind a responsive UI.

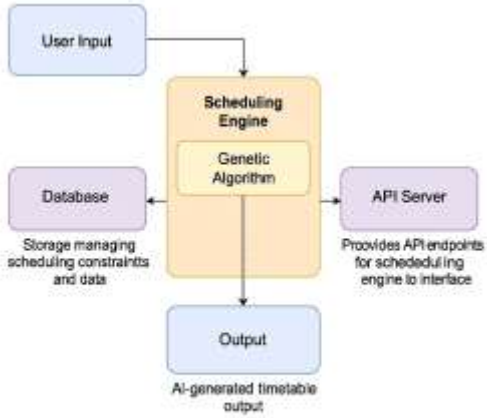

Figure 1: Block Diagram of Adaptive Scheduler

Figure 2: Genetic Algorithm Workflow

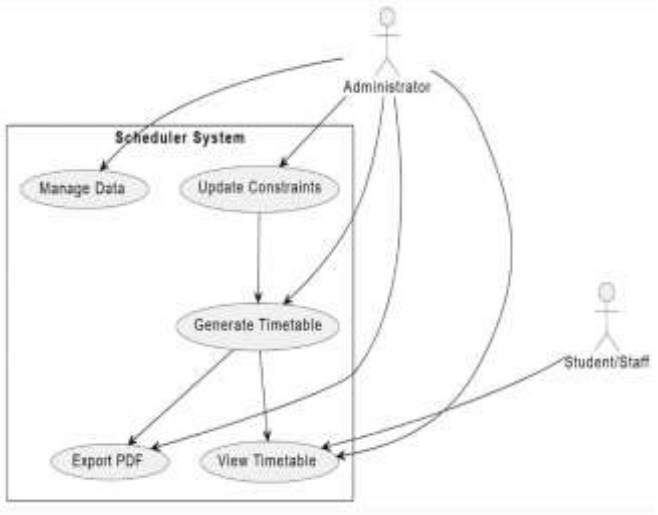

Figure 3: Use Case Diagram

IV. IMPLEMENTATION

The implementation phase involved transforming the theoretical design and methodology into a functional software system. The project was built using a modular architecture, separating core algorithmic logic, backend services, and the frontend interface for effective maintenance and scalability. This section elaborates on the chosen tech stack, backend structure, frontend integration, and the development tools employed.

A. Technology Stack

The project utilized a combination of technologies tailored to meet the specific functional and integration requirements of the system. Python was employed for algorithm development and backend services, while Flask served as the backend framework for building RESTful APIs. The frontend interface was developed using HTML, CSS, and JavaScript to render and interact with the timetable. Postman was used to test and validate backend API endpoints, and XAMPP (Cross-Platform (X), Apache (A), MariaDB (M), PHP (P), and Perl (P)) provided the local server environment for handling server-side processes during development. JSON was adopted as the data format for seamless communication between frontend and backend. For data visualization, JavaScript-powered dynamic HTML tables were implemented, and a backend route in Python was used to generate downloadable PDF versions of the timetables. Each tool played a crucial role in ensuring smooth functionality and effective integration across all components of the system.

B. Backend Architecture and Logic Flow

The backend was structured using Flask, where various components were logically separated for better maintainability. The core genetic algorithm and timetable optimization logic resided in a Python script named timetable.py. This file contained functions for selection, crossover, mutation, and fitness evaluation. The primary Flask application, defined in app.py, managed routing and request handling. It exposed endpoints for generating timetables and downloading PDFs. All initial data—including faculty, subjects, and schedules—were loaded from structured JSON files using a dedicated module. This architecture ensured clean separation of algorithm, data, and server logic, allowing efficient debugging and future scalability.

C. API Endpoint Design

The communication between the frontend and backend was facilitated through well-defined API endpoints. The primary endpoint, POST /generate-timetable, received input parameters such as faculty names, subjects, and constraints from the frontend, triggered the timetable generation process, and returned the result in a structured JSON format. Another endpoint, GET /download-pdf, enabled users to download the generated timetable in PDF format. These APIs were tested and validated using Postman to ensure that all inputs and responses were handled correctly and that error scenarios were gracefully managed. This RESTful approach enabled seamless integration and smooth user interactions.

D. Frontend Integration and Display

The frontend interface was designed to be lightweight, responsive, and easy to use. Users could enter required input details through form fields, and upon submission, JavaScript sent a request to the backend and rendered the resulting timetable dynamically on the page. The timetable was displayed in a tabular format, with proper styling to differentiate between days, time slots, and subjects. JavaScript was responsible for parsing the JSON response and updating the DOM. Additionally, a button was provided to export the timetable as a PDF, invoking the relevant backend endpoint. This frontend-backend synchronization provided a real-time experience for users.

E. Testing and Debugging

Testing was carried out at multiple stages of development. Unit tests were written for key algorithm components such as fitness calculation, mutation, and conflict detection. The integration between backend APIs and frontend inputs was tested manually and using Postman to ensure robust communication. Performance testing was also performed to ensure the timetable generation process completed within acceptable time frames for different input sizes. Browser developer tools were used to track frontend issues and ensure data was rendered accurately. Overall, a thorough testing strategy helped refine the system and minimize bugs.

V. RESULT AND DISCUSSION

The Adaptive Scheduler application successfully demonstrates the ability to dynamically generate optimized academic timetables based on user-defined data inputs and constraints. The results show a structured, conflict-free schedule that balances faculty availability, subject distributions, and time slots across different weekdays. Users can generate as shown in and visually view the AI-optimized timetable as shown in Fig. 4, Fig. 5 and Fig. 6 with a single click and also download it as a PDF for offline access or distribution. The frontend effectively displays the timetable using HTML tables, while the backend handles the scheduling logic and ensures constraint satisfaction. Although no numerical accuracy metrics were calculated, qualitative evaluation through interface testing and user review indicated that the schedules produced were correct, practical, and in alignment with academic needs.

The primary advantage of this system lies in its adaptability and automation. Unlike static scheduling systems or manual methods, the Adaptive Scheduler leverages AI-driven logic to intelligently assign time slots, reducing human effort and minimizing conflicts. The ability to quickly generate and download a PDF version of the schedule adds convenience and usability. Additionally, the use of technologies such as Flask, JavaScript, and jsPDF allows for cross-platform compatibility and seamless integration. Its modular design also means future enhancements, such as personalized student or faculty views, can be easily integrated.

Despite its strengths, the system has certain limitations. The scheduling logic primarily focuses on generating a general-purpose timetable rather than department-specific or student-specific versions. The system also does not currently account for sudden changes such as leave requests, emergency substitutions, or classroom unavailability, which are typical in real-world scenarios. Moreover, the timetable is generated based on pre-defined constraints and datasets, so inaccuracies in the input data may lead to flawed outputs.

During the development of this project, several challenges emerged. Designing an efficient fitness function that could evaluate a wide range of constraints without increasing computational complexity required considerable effort. Integrating the frontend with the backend, especially in managing asynchronous fetch requests and dynamically updating the timetable, also posed technical hurdles. Debugging API endpoints and ensuring compatibility between various tools (e.g., XAMPP for local hosting and Postman for API testing) added additional complexity. However, through iterative testing and refinements, these issues were systematically addressed.

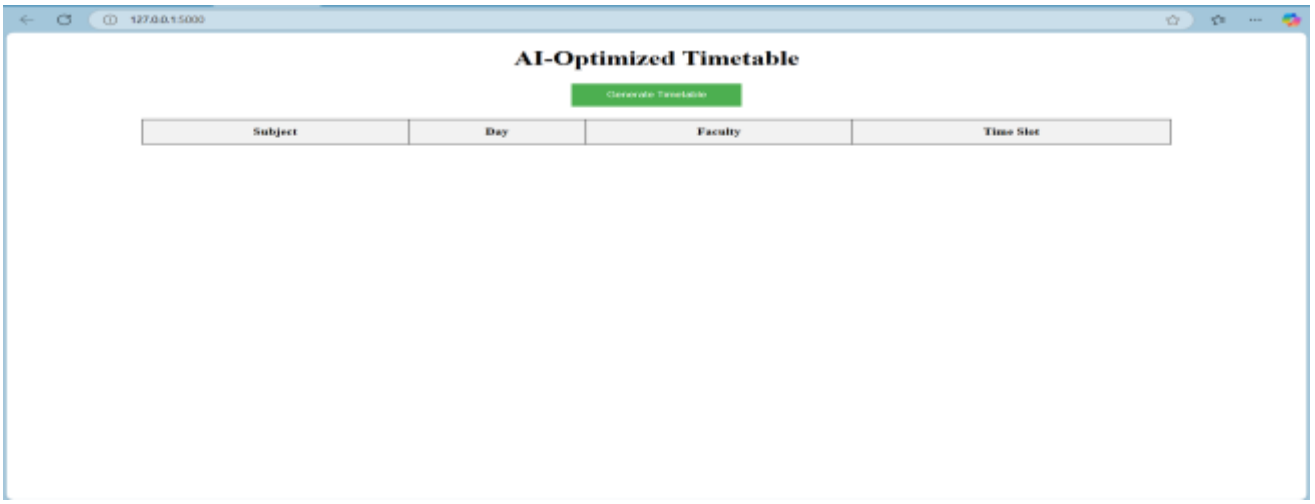

Figure 4: Web Application Interface

Figure 5: AI Optimized Timetable

AI-Optimized Timetable

Subject	Day	Faculty	Time Slot
OS	Monday	Prof. Walikar	9:00 AM - 10:30 AM
SE	Tuesday	Prof. Varalakshmi	2:00 PM - 3:30 PM
ML	Thursday	Prof. Rubini	9:00 AM - 10:30 AM
DSA	Thursday	Prof. Manjunath	11:00 AM - 12:30 PM
Algorithms	Friday	Prof. Vijayakumari G	9:00 AM - 10:30 AM
CN	Friday	Prof. Walikar	11:00 AM - 12:30 PM
AI	Friday	Prof. Rubini	2:00 PM - 3:30 PM

Figure 6: PDF View of AI Optimized Timetable

VI. CONCLUSION AND FUTURE WORK

In Conclusion, The Adaptive Scheduler system provides a practical and efficient solution for the complex problem of academic timetable generation. Leveraging Genetic Algorithms for constraint-based optimization, the system simplifies the traditionally manual and error-prone process by intelligently allocating classes, rooms, and instructors. The project meets essential requirements such as handling faculty preferences, room constraints, and subject loads, while offering features like real-time generation, PDF export, and visual representations through a user-friendly web interface. This AI-driven solution proves to be scalable, adaptable, and reliable for academic institutions. The successful implementation demonstrates the viability of using evolutionary algorithms in education planning tools and reflects the potential of automation in administrative processes.

While the current version fulfills its core objectives, several enhancements can be considered for future development. One significant direction is the integration of machine learning for predictive timetable suggestions based on historical usage patterns and academic calendars. Additionally, implementing real-time editing features and dynamic constraint management would allow more flexibility during live scheduling scenarios. Security features like user authentication and access control can be added to restrict unauthorized modifications. Expanding the platform into a full-fledged role-based dashboard for students, faculty, and administrators will further improve user experience. Other potential improvements include integrating Google Calendar sync, developing a mobile version of the platform, and enabling multi-semester planning features. These improvements will make the system more robust, personalized, and aligned with evolving academic needs.

VII. ACKNOWLEDGMENT

Authors gratefully acknowledge CMR University for providing the necessary facilities and support to carry out this work. Special appreciation is reserved for our project mentor for their valuable contributions throughout the project development. We also extend our sincere thanks to the reviewers for their insightful suggestions and constructive feedback.

REFERENCES

[1] Ghasemi, P. Moradi and M. Fathi, "Integrating ABC with genetic grouping for university course timetabling problem," 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 2015, pp. 24-29, doi: 10.1109/ICCKE.2015.7365857.
[2] S. Yang and S. N. Jat, "Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 1, pp. 93-106, Jan. 2011, doi: 10.1109/TSMCC.2010.2049200.
[3] S. K. Siledar and V. B. Musande, "Smart Time Table Generation using Artificial Intelligence," Grenze International Journal of Engineering and Technology, vol. 9, no. 2, June 2023, Grenze ID: 01.GIJET.9.2.336.
[4] N. G. A. P. H. Saptarini, P. I. Ciptayani, N. W. Wisswani and I. W. Suasnawa, "Adaptive Genetic Algorithm for High School Time-Table," Journal of Physics: Conference Series, vol. 1569, 2020, doi: 10.1088/1742-6596/1569/3/032095.
[5] D. M. Premasiri, "University Timetable Scheduling Using Genetic Algorithm Approach: Case Study: Rajarata University of Sri Lanka," Journal of Engineering Research and Application, vol. 8, issue 12 (Part-II), Dec. 2018, pp. 30-35. ISSN: 2248-9622.
[6] D. Srinivasan, T. H. Seow, and J. X. Xu, "Automated Time Table Generation Using Multiple Context Reasoning for University Modules," Proceedings of the International Conference, Department of Electrical & Computer Engineering, National University of Singapore, 2002, IEEE Catalog Number: 0-7803-7282-4/02.
[7] K. N. Subang, E. I. Balaba, and J. C. Agoylo Jr., "Optimizing Course Scheduling with Genetic Algorithms: A Dynamic Approach," SAR Journal, vol. 7, issue 4, pp. 296–302, Dec. 2024, doi: 10.18421/SAR74-02. ISSN: 2619-9955.
[8] Jawale, R. Sabale, and D. Kulkarni, "Automated timetable generator system," Int. J. Creative Res. Thoughts, vol. 3, June 2021.
[9] M. V. Rane, V. M. Apte, V. N. Nerkar, M. R. Edinburgh, and K. Y. Rajput, "Automated timetabling system for university course," Proc. Int. Conf. Emerging Trends Eng. Technol., Mar. 2021.
[10] S. Thakare and T. Nikram, "Automated timetable generation using genetic algorithm," Int. J. Creative Res. Thoughts, vol. 9, July 2020.
[11] K. Herath, "Genetic algorithm for university course timetabling problem," Electron. Theses Dissertations, Univ. of Mississippi, 2017. [Online]. Available: https://egrove.olemiss.edu/etd/443
[12] K. A. Downslund, "Simulated annealing solutions for multi-objective scheduling and timetabling," in Modern Heuristic Search Methods, Chichester, U.K.: Wiley, 1996, pp. 155–166.
[13] Hertz, "Tabu search for large scale timetabling problems," Eur. J. Oper. Res., vol. 54, pp. 39–47, 1991.
[14] D. F. Dofadar, R. H. Khan, S. Hasan, T. A. Taj, A. Shakil, and M. Majumdar, "A hybrid evolutionary approach to solve university course allocation problem," arXiv preprint arXiv:2212.02230, 2022.
[15] Ruddick, E. Genov, L. R. Camargo, T. Coosemans, and M. Messagie, "Evolutionary scheduling of university activities based on consumption forecasts to minimize electricity costs," arXiv preprint arXiv:2202.12595, 2022.
[16] "AI powered timetable scheduler and management," J. Emerging Technol. Innovative Res., vol. 10, no. 5, Oct. 2024.
[17] "Automatic timetable generation system," Int. J. Creative Res. Thoughts, vol. 9, no. 10, Oct. 2023
[18] "Automated timetable generation for academic institutions," AIP Conf. Proc., vol. 3075, no. 1, 020094, 2023.
[19] "University schedule generator," AIP Conf. Proc., vol. 3156, no. 1, 040002, 2023
[20] "FET (Free Timetabling Software)," Wikipedia.