# Advanced Timetable Generation Solution for Educational Institutes

Udayantha D.M.S
(IT21266164)

B.Sc. (Hons) Degree in Information Technology Specializing in Information Technology

Department of Information Technology
Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

# Research and Develop an Effective Timetable Evaluation method

Udayantha D.M.S

(IT21266164)

Dissertation submitted in partial fulfilment of the requirements for the Bachelor of Science Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

## Declaration

To the best of our knowledge and belief, this report does not contain any material that has already been published or authored by someone else, unless it is expressly recognized in the text. As a result, I hereby affirm that this is entirely my own work and that no previously submitted materials for a degree or certificate at any other university or higher education institution have been used in this proposal without my consent.

| Name | Student ID | Signature |
|---|---|---|
| Udayantha D.M S | IT21266164 | Sasitha |

## Abstract

This project aims to develop a Comprehensive Timetable Evaluation System designed to assess and optimize university timetables generated by various algorithms such as Genetic Algorithms, Colony Optimization, and Reinforcement Learning. The proposed system evaluates the quality of timetables based on multiple criteria, including the fulfillment of hard and soft constraints, resource utilization, workload balance, and institutional priorities. By providing clear, quantifiable metrics and scores, the system supports decision-makers in selecting the most effective timetables, improving scheduling processes, and optimizing resource management.

The system will incorporate advanced evaluation techniques to strictly penalize violations of hard constraints, such as scheduling conflicts, while applying flexible penalties for soft constraint violations, such as uneven workloads or gaps in schedules. Additionally, the system will assess resource utilization efficiency, measure schedule compactness, and prioritize key courses to ensure optimal time slot allocation. The scoring mechanism will integrate multiple objectives, offering a holistic assessment of each timetable's performance.

A critical aspect of the project is scalability, enabling the system to handle large datasets and complex schedules while maintaining high performance. Flexibility will also be emphasized, allowing the scoring weights to be adjusted to reflect shifting institutional goals. Furthermore, the system will feature a chatbot interface, enabling students and lecturers to easily access lecture times and locations, enhancing user engagement and streamlining information retrieval.

The project's ultimate goal is to provide universities with a data-driven, objective approach to timetable evaluation, enhancing decision-making, promoting fairness in workload distribution, and ensuring efficient use of resources in the scheduling process.

## Contents

## List of Figures

# 1. INTRODUCTION

## 1.1 Background Literature

### Background

University timetabling is a critical operational challenge faced by academic institutions globally. The task of creating a timetable involves assigning times and locations to a set of courses while satisfying a variety of constraints. These constraints include room capacity, instructor availability, course prerequisites, and preventing schedule conflicts for students. Traditionally, timetable creation has been a manual process or supported by simple rule-based systems, which are time-consuming and prone to errors.

With the increasing complexity of modern university scheduling, there is a growing need for more sophisticated approaches that not only generate feasible timetables but also optimize them according to various objectives such as efficient resource utilization, workload balance, and student preferences.

Algorithmic methods such as Genetic Algorithms (GA), Colony Optimization (CO), and

Reinforcement Learning (RL) have emerged as viable solutions for addressing this problem. However, while these algorithms can generate timetables, there is still a lack of systems designed to comprehensively evaluate the quality of these schedules. Existing approaches often focus on generating feasible timetables but do not provide robust mechanisms for evaluating their performance based on institutional objectives, constraints, and user preferences. This gap highlights the need for a Comprehensive Timetable Evaluation System that can objectively score and compare timetables produced by various algorithms, allowing decision-makers to choose the most effective option.

## Literature Survey

The generation and evaluation of university timetables are longstanding problems in operational research, often involving the complex balancing of hard and soft constraints, resource management, and stakeholder preferences. While many optimization algorithms have been explored for generating timetables, recent studies emphasize the importance of evaluating the quality of these timetables systematically. In this context, the evaluation process is as crucial as the generation process, especially when employing algorithms such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Reinforcement Learning (RL).

1. Genetic Algorithms (GA) for Timetable Generation

Genetic Algorithms have been widely applied to university timetabling due to their effectiveness in handling multi-objective problems. As demonstrated in several studies [1], GA's evolutionary approach makes it highly suitable for timetable generation, where constraints such as class schedules, room availability, and instructor assignments must be respected. GA optimizes timetables by evolving a population of solutions, with each iteration bringing the schedule closer to an optimal state. However, its reliance on random processes can sometimes result in suboptimal solutions, necessitating robust evaluation methods to identify and score the best results accurately.

2. Ant Colony Optimization (ACO)

Ant Colony Optimization is inspired by the behavior of ants seeking the shortest paths to food and has been effectively adapted for timetabling problems [2]. ACO excels in finding optimal solutions in dynamic environments, making it particularly useful for timetables that need to adapt to last-minute changes or evolving constraints. ACO's ability to reinforce better solutions over time allows it to effectively manage complex scheduling problems. However, much like GA, ACO-generated timetables require a rigorous evaluation framework to assess their feasibility and adherence to institutional objectives.

3. Reinforcement Learning (RL)

Reinforcement Learning is gaining traction in the area of timetable scheduling due to its ability to learn from interactions with the environment and optimize decisions over time [3]. RL models scheduling as a sequential decision-making problem, where the algorithm progressively improves its actions to maximize a reward function. This approach allows RL to adapt timetables dynamically as conditions change, making it a promising technique for educational institutions with flexible scheduling needs. Nonetheless, the performance of RLgenerated timetables must be evaluated comprehensively to ensure that they meet hard and soft constraints, resource utilization goals, and organizational objectives.

4. Evaluation Methods for Timetable Quality

In timetabling research, the evaluation of generated schedules is just as important as their creation. Timetables must be scored not only based on their ability to satisfy hard constraints, such as avoiding scheduling conflicts, but also on their performance in soft constraint categories, such as minimizing gaps in students' schedules or balancing instructors' workloads. According to research by Kingston [4], evaluation metrics must also consider resource utilization, compactness, and institutional priorities. A multi-objective scoring system that integrates these factors is crucial for distinguishing between good and excellent timetables, particularly when using heuristic or metaheuristic algorithms like GA, ACO, and RL.

5. Hybrid Approaches in Timetable Scheduling

The combination of different optimization algorithms, commonly referred to as hybrid approaches, has been explored extensively in recent literature [5]. These models integrate the strengths of multiple algorithms, such as GA and ACO or RL and ACO, to overcome individual limitations. For instance, GA may generate an initial set of timetables, which ACO or RL can then refine. Hybrid models are particularly effective in improving the robustness and flexibility of timetable generation processes. However, their success hinges on the quality of the evaluation systems that score the resulting timetables.

6. Scoring Mechanisms

Scoring mechanisms are essential for evaluating the quality of timetables generated by different algorithms. These mechanisms assign a numerical score to each timetable based on various criteria, including constraint satisfaction, resource utilization, and overall coherence. As highlighted by MirHassani [6], scoring systems should be multi-dimensional, incorporating penalties for hard constraint violations and rewards for meeting specific objectives, such as minimizing idle time or maximizing room usage. Such systems are critical for comparing the performance of different algorithms and selecting the best timetable based on institutional needs.

Conclusion

The literature consistently supports the use of Genetic Algorithms, Ant Colony Optimization, and Reinforcement Learning for university timetable generation. However, the effective implementation of these algorithms relies heavily on the development of comprehensive evaluation systems. These systems must assess the generated timetables on multiple dimensions, including hard and soft constraint satisfaction, resource utilization, and alignment with institutional objectives. By integrating multi-objective scoring mechanisms, the evaluation system can provide administrators with clear, actionable insights into the quality of each timetable, ultimately supporting more informed decision-making in scheduling.

**Relevant Studies and Techniques to Master**

1. **Hybrid Algorithm Approaches:**

   - **Genetic Algorithms (GA):** Understand the fundamentals and advanced techniques in genetic algorithms, including selection, crossover, mutation, and fitness evaluation. Key readings include **"Genetic Algorithms in Search, Optimization, and Machine Learning"** by John Holland.
   - **Reinforcement Learning (RL):** Gain proficiency in RL algorithms, such as Qlearning and policy gradients, focusing on their application to

scheduling problems. **"Reinforcement Learning: An Introduction"** by Richard Sutton and Andrew Barto is a key resource.

- **Colony Optimization:** Study the principles and implementations of ant colony optimization (ACO) for scheduling. **"Ant Colony Optimization"** by Marco Dorigo and Thomas Stützle provides comprehensive coverage of this technique.

2. **Constraint Handling:**

   - **Constraint Satisfaction Problems (CSP):** Learn about constraint satisfaction and optimization techniques to effectively handle hard and soft constraints in timetable scheduling. **"Constraint Processing"** by Rina Dechter is recommended for understanding CSP fundamentals.

3. **Scoring and Evaluation Metrics:**

   - **Timetable Scoring Systems:** Study methods for evaluating timetable quality, including scoring based on hard and soft constraints, resource utilization, and compactness. **"Evaluation Metrics for University Timetable Scheduling"** by Xiang-Hua Li and Zhi-Wei Li covers these aspects.

4. **Algorithm Integration and Hybrid Approaches:**

   - **Combining Algorithms:** Master techniques for integrating genetic algorithms, reinforcement learning, and colony optimization to create a hybrid scheduling approach. Look into literature on hybrid metaheuristics, such as **"Hybrid Metaheuristics: Research, Development and Applications"** edited by Michel Gendreau and Jean-Yves Potvin.

5. **Resource Utilization and Optimization:**

   - **Resource Allocation:** Study strategies for optimizing the use of resources such as rooms and instructors. Techniques for efficient space usage and workload balancing are critical.

6. **User Feedback Incorporation:**

   - **Incorporating User Feedback:** Learn methods for integrating user feedback into the scheduling system to improve its effectiveness and user satisfaction.

7. **Chatbot Integration:**

   - **Developing Chatbots:** Understand how to build and integrate chatbots for interacting with the timetable system. Explore chatbot frameworks and integration techniques for user-friendly interfaces.

8. **Scalability and Performance Optimization:**

   - **System Scaling:** Study techniques for scaling your system to handle large datasets and complex scheduling scenarios. Focus on performance optimization strategies to ensure efficient operation.

## 1.2 Research Gap

| **Area of Study** | **Current Knowledge** | **Research Gap** |
|---|---|---|
| Algorithms | Genetic Algorithms (GA), Reinforcement Learning (RL), and Colony Optimization (CO) have been individually studied for scheduling problems. | Insufficient focus on comparing the effectiveness of GA, RL, and CO in isolation for different aspects of timetable optimization. |

| | | |
|---|---|---|
| Constraint Handling | Methods for managing hard and soft constraints in scheduling are established, including constraint satisfaction and relaxation techniques. | Need for innovative approaches to handle complex and dynamic constraints in university timetabling, particularly integrating user-specific constraints. |
| Scoring and Evaluation Metrics | Various metrics exist for evaluating timetable quality, such as resource utilization and compactness. | Lack of comprehensive scoring systems that integrate multiple evaluation criteria (hard/soft constraints, resource utilization, etc.) for university timetables. |
| Chatbot Integration | Chatbots are used for various applications, and integration techniques are well-explored. | Limited research on integrating chatbots specifically with timetable evaluation systems to facilitate user interaction and constraint input. |
| Scalability and Performance | Scalability and performance optimization techniques are known in the context of general systems. | Need for strategies to scale and optimize performance specifically for complex timetable scheduling systems with large datasets. |

*Figure 1 reserch Gap*

## Research Gap

While numerous studies have applied optimization algorithms such as Genetic Algorithms (GA), Reinforcement Learning (RL), and Colony Optimization (CO) to the university timetabling problem, most research focuses on the individual application of these algorithms. There is insufficient comparative analysis to determine the relative strengths and weaknesses of these approaches across different dimensions of timetable optimization, such as resource utilization, conflict resolution, and schedule compactness. This creates a gap in understanding which algorithm performs best under specific institutional scenarios or constraints.

Moreover, although methods for handling hard and soft constraints are well-established, including rule-based and constraint satisfaction techniques, existing systems often fail to accommodate complex, evolving, or user-defined constraints. Real-world university environments frequently involve dynamic factors, such as instructor preferences, student grouping rules, or special accessibility requirements, which are often neglected or oversimplified in current solutions.

In terms of timetable evaluation, current systems typically rely on a limited set of metrics and lack a comprehensive scoring framework that integrates multiple factors—such as constraint violations, resource usage, workload distribution, and user preferences—into a unified evaluation mechanism. This limits the ability of administrators to objectively compare the quality of different timetables generated by various algorithms.

Furthermore, while chatbot technologies are widely used across multiple domains, their integration into university timetable systems—especially for evaluation and user interaction—remains underexplored. There is a clear opportunity to enhance usability by enabling students and lecturers to provide feedback or input constraints directly through conversational interfaces.

Lastly, as universities grow and datasets become more complex, scalability and system performance become critical. Current literature does not sufficiently address performance optimization strategies tailored to large-scale timetable evaluation systems. Effective methods are needed to ensure that these systems can maintain accuracy and responsiveness even as the volume of data and number of constraints increase.

Together, these gaps highlight the need for a **Comprehensive Timetable Evaluation System** that not only compares different algorithmic outputs but also handles diverse constraints, supports user interaction, and performs efficiently at scale.

## 1.3 Research Problem

**Research Problem Statement**

"How can a comprehensive evaluation framework be developed to objectively assess university timetables generated by different algorithms—namely Genetic Algorithms, Reinforcement Learning, and Colony Optimization—based on key metrics such as constraint satisfaction, resource utilization, compactness, and overall schedule quality?"

### *1. Algorithm Comparison for Timetable Evaluation*

Evaluating the effectiveness of different algorithms—**Genetic Algorithms**, **Reinforcement Learning**, and **Colony Optimization**—in generating university timetables requires a consistent and fair comparison framework. The challenge lies in assessing each algorithm's ability to handle scheduling constraints, optimize resource allocation, and produce high-quality timetables. This involves establishing **uniform evaluation metrics** and methodologies that can objectively compare the outcomes of each algorithm across various dimensions of timetable performance.

### *2. Comprehensive Scoring System Development*

Designing a robust and flexible **scoring system** is critical for objectively evaluating timetables. The system must account for multiple evaluation criteria, such as:

- **Hard and soft constraint satisfaction**
- **Resource utilization efficiency**
- **Schedule compactness and balance**
- **Priority handling for key modules or instructors**

The core challenge is to develop a scoring mechanism that integrates these metrics into a **unified model** capable of assigning meaningful and interpretable scores. This enables clear performance comparisons across different algorithm-generated schedules and supports data-driven decision-making in timetable selection.

### *3. Chatbot Integration as a Support Module*

Although not central to the research, integrating a **chatbot interface** into the timetable system adds value by improving user interaction and accessibility. The chatbot must:

- Access and present **real-time timetable data**
- Respond to queries about **lecture times, room allocations, and instructor details**
- Allow students and lecturers to **submit preferences or constraints** (optional)
- Deliver a **user-friendly experience** while maintaining system performance

The key challenge is ensuring smooth **backend integration**, so the chatbot remains responsive, accurate, and aligned with the core timetable data and evaluation logic.

## 1.4 Research Objectives

### Main Objective

- **To develop a comprehensive system for evaluating and comparing university timetables generated by different algorithms**
  The main goal of this research is to create a robust evaluation framework that objectively assesses timetables generated by Genetic Algorithms (GA), Reinforcement Learning (RL), and Colony Optimization (CO) based on key performance metrics. The system will focus on constraints satisfaction, resource utilization, and overall quality, enabling effective decision-making in timetable scheduling.

## Sub-Objectives

1. **To design a comprehensive evaluation framework**
   - Create a framework that includes defined criteria for evaluating timetables, such as hard and soft constraint satisfaction, resource utilization, compactness, and alignment with organizational objectives.
2. **To compare the effectiveness of different scheduling algorithms (GA, RL, CO)**
   - Evaluate the performance of Genetic Algorithms, Reinforcement Learning, and Colony Optimization in generating timetables based on key metrics like resource optimization and constraint satisfaction.
3. **To develop a robust scoring mechanism**
   - Design and implement a scoring system that integrates various evaluation metrics to objectively quantify the effectiveness of timetables, balancing multiple factors for a comprehensive score.
4. **To assess the adaptability and scalability of the evaluation system**
   - Ensure that the system is adaptable to different types of scheduling problems and scalable to accommodate future changes, such as larger institutions or evolving scheduling needs.
5. **To provide insights for optimizing timetable generation**
   - Analyze the results from different algorithms and provide insights and recommendations on how timetables can be optimized for better resource allocation and scheduling efficiency.

# 2. Methodology

## 2.1 Methodology

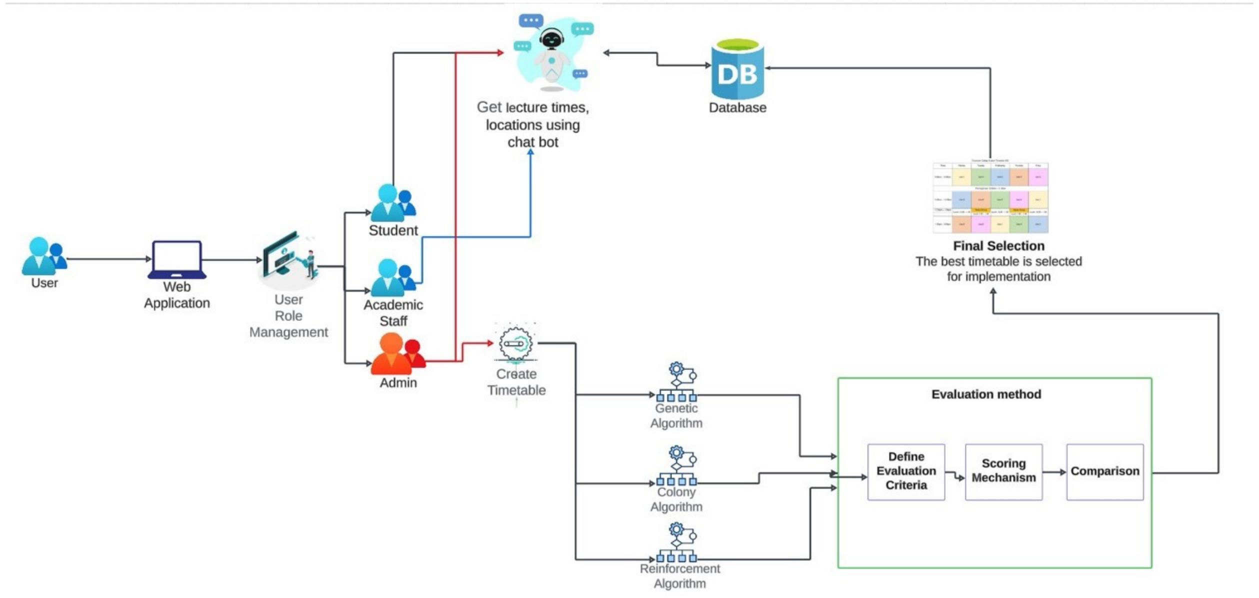


*Figure 2 system Overview*

## System Overview

The **University Timetable Evaluation System** is designed to provide a comprehensive and robust framework for evaluating and scoring university timetables generated by various scheduling algorithms. This system aims to enhance decision-making in timetable management by offering detailed and accurate performance metrics, improving resource allocation, and optimizing scheduling. Additionally, the system integrates a chatbot interface to provide user interactions, facilitating easier access to timetable-related information.

---

## 1. System Components

1. **Algorithm Evaluation Module**
   - **Purpose**: Evaluate timetables generated by different algorithms, including **Genetic Algorithms (GA)**, **Reinforcement Learning (RL)**, and **Colony Optimization (CO)**.

    - **Functionality**: This module compares the performance of timetables based on metrics such as **constraint satisfaction**, **resource utilization**, and **overall timetable quality**. It ensures a detailed and comprehensive analysis of the strengths and weaknesses of each algorithm.
2. **Scoring System**
    - **Purpose**: Provide an objective and thorough assessment of timetables.
    - **Functionality**: Integrates multiple evaluation criteria, including **hard and soft constraints**, **resource usage**, and **timetable compactness**. This system quantifies and balances these factors to provide accurate scores that reflect the quality of timetables.
3. **Constraint Handling and Evaluation**
    - **Purpose**: Effectively manage and evaluate constraints imposed on timetables.
    - **Functionality**: Incorporates both **hard and soft constraints**, identifies violations, and evaluates their impact on timetable quality. This module ensures that all constraints are properly integrated into the scoring process, leading to a fair evaluation of the timetables.
4. **Dynamic Adaptability Assessment**
    - **Purpose**: Assess the adaptability of timetables to real-time changes and disruptions.
    - **Functionality**: Evaluates how well timetables adapt to unexpected changes, such as alterations in room or instructor availability. This module ensures the system's flexibility, allowing timetables to adjust dynamically to evolving conditions.
5. **User Feedback Integration**
    - **Purpose**: Incorporate user feedback to refine and enhance the evaluation process.
    - **Functionality**: Develops mechanisms to collect and integrate user feedback into the evaluation system, adjusting scores based on practical needs, user satisfaction, and system performance. This ensures the system remains user-centered and responsive to real-world requirements.
6. **Chatbot Interface**
    - **Purpose**: Provide user interaction through a chatbot to access lecture times, locations, and related information.

   - **Functionality**: Implements a chatbot that integrates seamlessly with the timetable management system to handle queries from students and lecturers. It ensures users receive accurate, timely responses, making it easier to interact with the system for schedule-related queries and updates.

7. **System Performance and Scalability**
   - **Purpose**: Ensure the system can efficiently handle large datasets and complex scheduling scenarios.
   - **Functionality**: Implements strategies for optimizing system performance and scalability. This includes load balancing, data management, and testing to maintain high performance and reliability as the system grows.

---

## 2. System Architecture

The system architecture consists of several interconnected components, each responsible for specific functions within the timetable evaluation process:

- **Data Input Module**: Collects and processes timetable data, user constraints, and feedback.
- **Algorithm Processing Module**: Handles the evaluation of timetables generated by different algorithms (GA, RL, CO).
- **Scoring and Analysis Engine**: Computes scores based on the evaluation criteria and generates detailed reports on timetable quality.
- **Chatbot Integration Layer**: Connects the chatbot interface with the timetable management system for real-time user interactions.
- **User Interface**: Provides a user-friendly interface for accessing timetable information, submitting feedback, and interacting with the chatbot.
- **Backend Infrastructure**: Ensures robust performance, scalability, and data management, ensuring the system can handle large datasets and complex scheduling needs.

---

## 3. Workflow

The workflow of the University Timetable Evaluation System is designed to follow a logical sequence of operations:

1. **Timetable Generation**: Different algorithms generate timetables and submit them for evaluation.
2. **Evaluation**: The system evaluates the timetables based on predefined metrics, constraints, and evaluation criteria.
3. **Scoring**: Scores are calculated using the **scoring system** and **analysis engine**, ensuring a comprehensive evaluation of the timetables.
4. **Feedback Collection**: Feedback from users is collected and integrated into the evaluation process to refine future assessments.
5. **Interaction**: The **chatbot interface** facilitates real-time user interaction, providing information and answering queries related to the timetables.
6. **Reporting**: Detailed reports are generated, reflecting the timetables' overall quality, effectiveness, and suitability for the university's scheduling needs.

## 2.2 Commercialization aspect of the product

The **University Timetable Evaluation System** will be commercialized as a **cloud-based Software-as-a-Service (SaaS)** product, designed to meet the increasing demand for intelligent timetable quality assessment in universities and educational institutions. With academic institutions around the world adopting automated scheduling systems, there is a clear need for a robust evaluation tool that can validate, compare, and enhance the outputs of various scheduling algorithms—ensuring fairness, efficiency, and quality in academic timetables.

Initially, the system will be offered to both **state and non-state universities in Sri Lanka**, supporting academic administrators in making data-driven decisions when selecting or refining generated timetables. The system leverages advanced AI methodologies—**Genetic Algorithms (GA)**, **Reinforcement Learning (RL)**, and **Colony Optimization (CO)**—to analyze and score schedules based on multiple constraints, including resource utilization, hard and soft constraint satisfaction, compactness, workload balance, and adaptability.

The product will be offered in **tiered subscription plans** based on institution size, number of departments, and feature requirements such as custom constraint integrations, historical performance analytics, ERP integration, and chatbot capabilities. Universities will benefit from cost-effective access to a centralized evaluation dashboard with multi-user role access—allowing administrators, academic staff, and quality assurance teams to collaborate and review timetable effectiveness in real time.

The built-in **chatbot assistant** adds commercial value by improving user experience—allowing students and lecturers to query timetable scores, understand constraint violations, and view evaluation reports via natural conversation interfaces. This improves transparency and communication across the academic community.

In addition to universities, the product roadmap includes **expanding to schools and vocational institutions** that use third-party scheduling tools but lack a formalized method of evaluating schedule quality. Over time, the system can be customized for **enterprise applications** such as staff scheduling in **hospitals**, **call centers**, or **logistics companies**, where shift optimization and resource utilization are critical.

Through continuous updates and user-driven customization options, the system will position itself as a **pioneering solution in the timetable evaluation space**, supporting sustainable education reform and digital transformation. With its modular design, scalable cloud architecture, and AI-driven evaluation engine, this product is poised to become a **market leader** in timetable quality assurance and optimization.

### *1. Market Opportunity*

With universities and higher education institutions increasingly adopting automated scheduling systems, there's a significant gap in **timetable evaluation and quality assurance tools**. Most systems focus on generation but lack a standardized way to **evaluate the output across different algorithms**. This product addresses that unmet need by providing a robust, AI-powered **evaluation and scoring platform**—filling a critical gap in the academic scheduling process.

### *2. Product Positioning*

The University Timetable Evaluation System will be positioned as a **Software-as-a-Service (SaaS)** platform that provides real-time, multi-criteria evaluation of academic timetables. It will be marketed as a **decision-support tool** for academic administrators, QA committees, and IT departments who are responsible for scheduling, ensuring timetables are not only generated—but are also **measured, scored, and improved** continuously.

### *3. Target Market*

- **Primary Market:**
  - State universities
  - Non-state universities
  - Higher education institutions with complex scheduling needs
- **Secondary Market:**
  - Vocational training centers
  - Schools transitioning to digital timetable solutions
  - EdTech companies seeking integration with timetable generators

In the long term, the product can expand to **non-educational sectors** such as:

- **Healthcare** (e.g., nurse and doctor shift evaluation)
- **Project-based industries** (e.g., construction, IT, or service delivery)
- **Customer service centers** (e.g., staff roster evaluation)

### *4. Business Model*

The platform will adopt a **tiered subscription model**:

- **Basic Tier:** Limited number of users, evaluation of timetables with standard constraints and basic reports.
- **Professional Tier:** Advanced scoring, historical comparisons, user feedback integration, and priority support.
- **Enterprise Tier:** Full customization, API access for integration with ERP systems, advanced analytics dashboard, and chatbot interaction.

Additional monetization options:

- Custom module development (e.g., accreditation compliance checks)
- Consultancy services for algorithm benchmarking
- Data analytics packages for research or institutional reporting

### *5. Key Features that Drive Commercial Value*

- **Algorithm-Agnostic Evaluation:** Supports outputs from multiple algorithms like GA, RL, and CO—allowing universities to benchmark and select the best.
- **Multi-Criteria Scoring System:** Incorporates organizational priorities, hard/soft constraints, and compactness.
- **Integrated Chatbot:** Provides real-time access to scoring results, feedback mechanisms, and timetable insights for students and faculty.
- **User Role Management:** Customized access for admins, QA staff, academic heads, lecturers, and students.

- **Scalable Architecture:** Built to handle small departments or entire universities—supporting horizontal scalability in cloud infrastructure.

### *6. Competitive Advantage*

Unlike traditional scheduling platforms, this system doesn't focus on generation but rather on **ensuring quality, fairness, and performance** of already generated schedules. This gives it a **unique positioning** in the market and makes it **complementary rather than competitive** to existing scheduling software.

### *7. Go-to-Market Strategy*

- **Pilot Projects:** Partner with selected universities in Sri Lanka for pilot deployment and data collection.
- **Partnerships:** Collaborate with EdTech companies or existing timetable software vendors for integrations.
- **Academic Conferences & Journals:** Publish findings to create awareness in the academic community.
- **Workshops and Webinars:** Educate decision-makers on the importance of timetable evaluation and how it improves quality.

### *8. Future Expansion*

- Integration with **AI-driven auto-recommendation systems** for timetable improvements.
- Extension to **mobile platforms** for real-time access.
- Compatibility with **international scheduling standards** (for institutions abroad).
- Multi-language support for **regional and global scaling**.

## 2.3 Testing and Implementation

The implementation and testing phase is critical to ensure the accuracy, reliability, and performance of the University Timetable Evaluation System. This section outlines the systematic approach adopted to develop, deploy, and test the system's components, ensuring they meet functional and non-functional requirements.

The system was developed using a **modular architecture**, with clear separation of concerns between the frontend, backend, evaluation engine, and chatbot interface. The main technologies used include:

- **Frontend:** React.js with Vite, Tailwind CSS, Redux, and React Router
- **Backend:** Python with Flask/FastAPI, MongoDB
- **Evaluation Engine:** Pandas and Scikit-learn for constraint analysis and scoring logic
- **Reinforcement Learning Module:** TensorFlow/PyTorch integrated with OpenAI Gym
- **Chatbot:** Python-based interface with future Rasa integration for advanced capabilities

The Testing and Implementation phase of the **University Timetable Evaluation System** focused on developing and validating a dual-layered evaluation engine capable of assessing both hard and soft constraints with precision. The system was implemented using Python and tested through a structured methodology to ensure robustness, accuracy, and performance.

### *Implementation Overview*

The core of the system comprises two primary evaluation modules:

- **Hard Constraint Evaluation Function**
- **Soft Constraint Evaluation Function**

These functions were developed to work on dictionary-based timetable structures that reflect real-world scheduling entities such as rooms, activities, student groups, and lecturers. The functions collectively ensure that each timetable is not only conflict-free but also optimized for comfort, efficiency, and fairness.

---

### *2. Hard Constraint Evaluation Function*

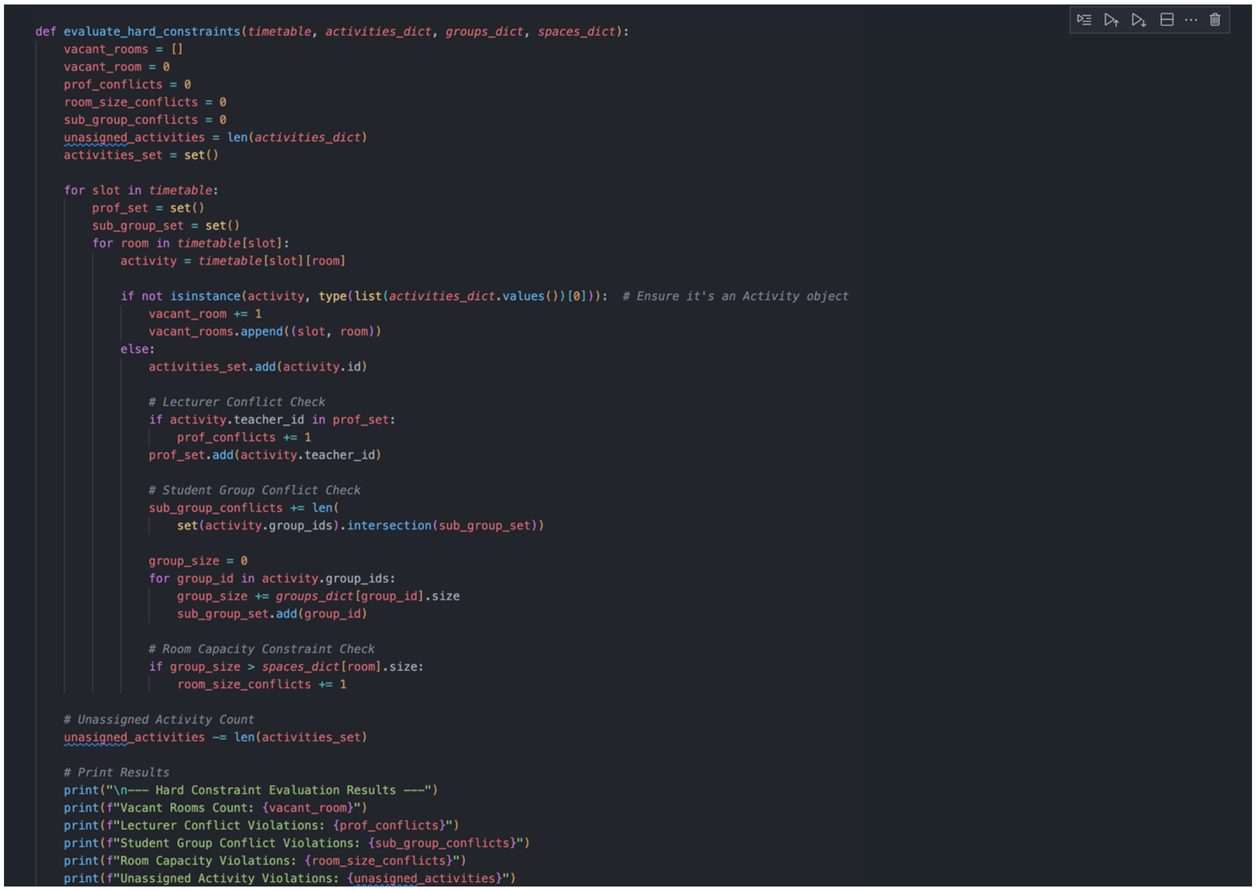
This function was implemented to assess **critical scheduling rules** that must not be violated:

- **Lecturer and Student Group Conflicts:** Detected using Python sets to check for overlapping assignments in a given timeslot.
- **Room Capacity Violations:** Validated by comparing the size of assigned student groups with the room's defined capacity.
- **Unassigned Activities:** Ensures every activity is placed in at least one timeslot.
- **Vacant Rooms:** Tracks inefficiencies in room utilization.

The function prints a detailed report summarizing:

- Vacant room count
- Lecturer double-bookings
- Student group scheduling conflicts
- Room overcapacity violations
- Unassigned activity count
- **Total hard constraint violation score**

This score helps quantify the **feasibility** of the timetable.



```python
def evaluate_hard_constraints(timetable, activities_dict, groups_dict, spaces_dict):
    vacant_rooms = []
    vacant_room = 0
    prof_conflicts = 0
    room_size_conflicts = 0
    sub_group_conflicts = 0
    unasigned_activities = len(activities_dict)
    activities_set = set()

    for slot in timetable:
        prof_set = set()
        sub_group_set = set()
        for room in timetable[slot]:
            activity = timetable[slot][room]

            if not isinstance(activity, type(list(activities_dict.values())[0])):  # Ensure it's an Activity object
                vacant_room += 1
                vacant_rooms.append((slot, room))
            else:
                activities_set.add(activity.id)

                # Lecturer Conflict Check
                if activity.teacher_id in prof_set:
                    prof_conflicts += 1
                prof_set.add(activity.teacher_id)

                # Student Group Conflict Check
                sub_group_conflicts += len(
                    set(activity.group_ids).intersection(sub_group_set))

                group_size = 0
                for group_id in activity.group_ids:
                    group_size += groups_dict[group_id].size
                    sub_group_set.add(group_id)

                # Room Capacity Constraint Check
                if group_size > spaces_dict[room].size:
                    room_size_conflicts += 1

    # Unassigned Activity Count
    unasigned_activities -= len(activities_set)

    # Print Results
    print("\n--- Hard Constraint Evaluation Results ---")
    print(f"Vacant Rooms Count: {vacant_room}")
    print(f"Lecturer Conflict Violations: {prof_conflicts}")
    print(f"Student Group Conflict Violations: {sub_group_conflicts}")
    print(f"Room Capacity Violations: {room_size_conflicts}")
    print(f"Unassigned Activity Violations: {unasigned_activities}")
```

*Figure 3 Hard constraint function code segment*

---

## 3. Soft Constraint Evaluation Function

The soft constraint module evaluates **quality-of-life** and **optimization** factors:

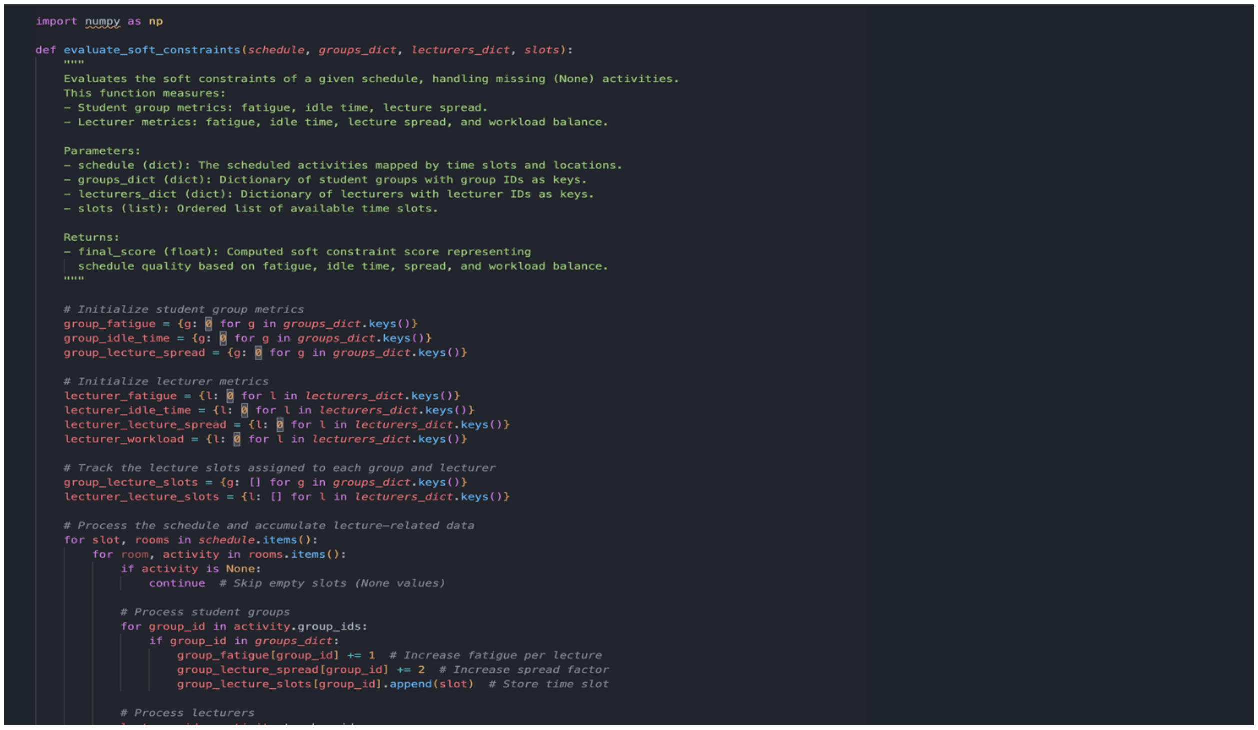
- **Student Metrics:** Fatigue, idle times between lectures, and scattered schedule patterns.
- **Lecturer Metrics:** Same as students, with added tracking of workload balance across staff.

Metrics were collected by looping through timeslots and calculating:

- Gaps in schedule (idle time)
- Total number of lectures

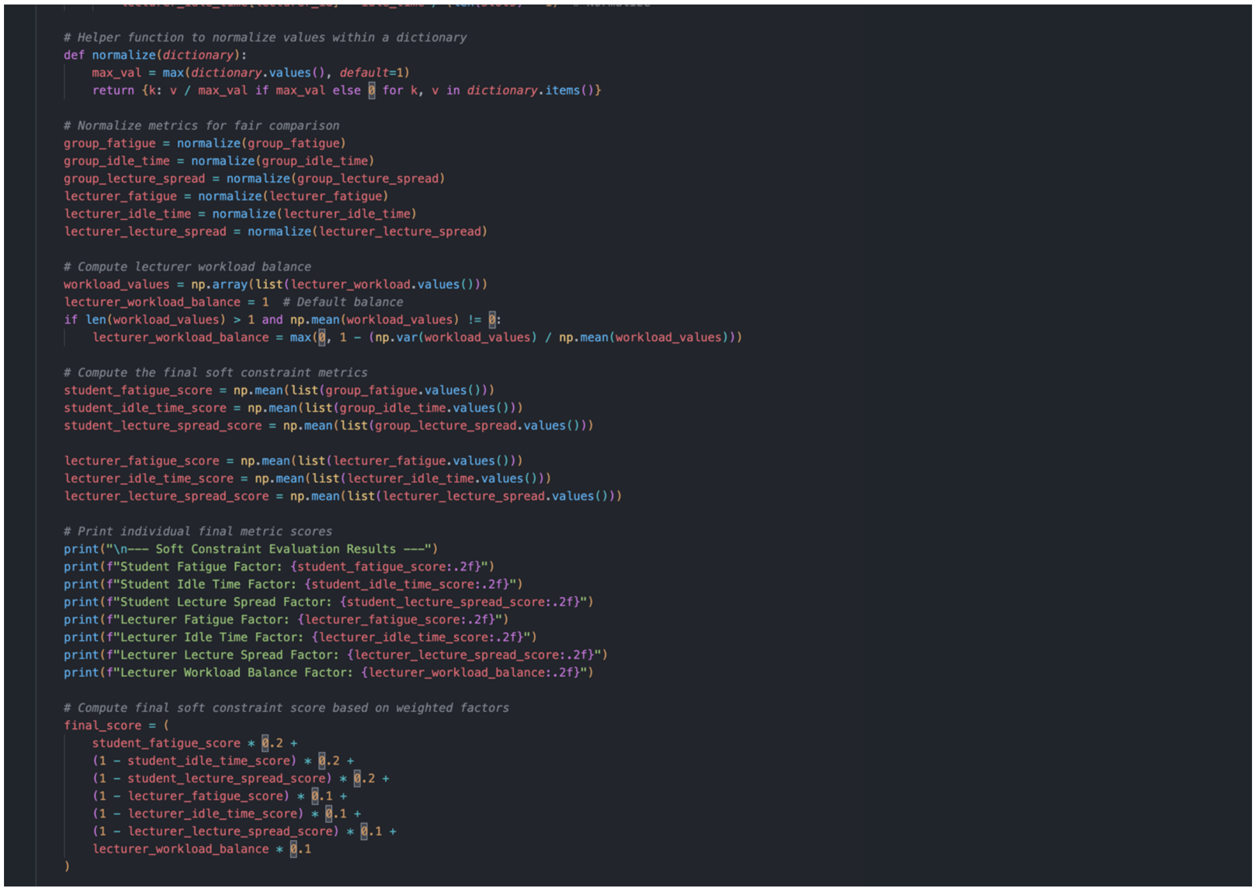
- Spread across available time
- Variance in lecturer workload

The final soft score is a **weighted combination** of all normalized factors, indicating the **overall quality** of the schedule from the perspective of resource utilization and user satisfaction.



```python
import numpy as np

def evaluate_soft_constraints(schedule, groups_dict, lecturers_dict, slots):
    """
    Evaluates the soft constraints of a given schedule, handling missing (None) activities.
    This function measures:
    - Student group metrics: fatigue, idle time, lecture spread.
    - Lecturer metrics: fatigue, idle time, lecture spread, and workload balance.

    Parameters:
    - schedule (dict): The scheduled activities mapped by time slots and locations.
    - groups_dict (dict): Dictionary of student groups with group IDs as keys.
    - lecturers_dict (dict): Dictionary of lecturers with lecturer IDs as keys.
    - slots (list): Ordered list of available time slots.

    Returns:
    - final_score (float): Computed soft constraint score representing
      schedule quality based on fatigue, idle time, spread, and workload balance.
    """

    # Initialize student group metrics
    group_fatigue = {g: 0 for g in groups_dict.keys()}
    group_idle_time = {g: 0 for g in groups_dict.keys()}
    group_lecture_spread = {g: 0 for g in groups_dict.keys()}

    # Initialize lecturer metrics
    lecturer_fatigue = {l: 0 for l in lecturers_dict.keys()}
    lecturer_idle_time = {l: 0 for l in lecturers_dict.keys()}
    lecturer_lecture_spread = {l: 0 for l in lecturers_dict.keys()}
    lecturer_workload = {l: 0 for l in lecturers_dict.keys()}

    # Track the lecture slots assigned to each group and lecturer
    group_lecture_slots = {g: [] for g in groups_dict.keys()}
    lecturer_lecture_slots = {l: [] for l in lecturers_dict.keys()}

    # Process the schedule and accumulate lecture-related data
    for slot, rooms in schedule.items():
        for room, activity in rooms.items():
            if activity is None:
                continue  # Skip empty slots (None values)

            # Process student groups
            for group_id in activity.group_ids:
                if group_id in groups_dict:
                    group_fatigue[group_id] += 1  # Increase fatigue per lecture
                    group_lecture_spread[group_id] += 2  # Increase spread factor
                    group_lecture_slots[group_id].append(slot)  # Store time slot

            # Process lecturers
```

*Figure 4 soft constrain function code segment*


```python
    # Helper function to normalize values within a dictionary
    def normalize(dictionary):
        max_val = max(dictionary.values(), default=1)
        return {k: v / max_val if max_val else 0 for k, v in dictionary.items()}

    # Normalize metrics for fair comparison
    group_fatigue = normalize(group_fatigue)
    group_idle_time = normalize(group_idle_time)
    group_lecture_spread = normalize(group_lecture_spread)
    lecturer_fatigue = normalize(lecturer_fatigue)
    lecturer_idle_time = normalize(lecturer_idle_time)
    lecturer_lecture_spread = normalize(lecturer_lecture_spread)

    # Compute lecturer workload balance
    workload_values = np.array(list(lecturer_workload.values()))
    lecturer_workload_balance = 1  # Default balance
    if len(workload_values) > 1 and np.mean(workload_values) != 0:
        lecturer_workload_balance = max(0, 1 - (np.var(workload_values) / np.mean(workload_values)))

    # Compute the final soft constraint metrics
    student_fatigue_score = np.mean(list(group_fatigue.values()))
    student_idle_time_score = np.mean(list(group_idle_time.values()))
    student_lecture_spread_score = np.mean(list(group_lecture_spread.values()))

    lecturer_fatigue_score = np.mean(list(lecturer_fatigue.values()))
    lecturer_idle_time_score = np.mean(list(lecturer_idle_time.values()))
    lecturer_lecture_spread_score = np.mean(list(lecturer_lecture_spread.values()))

    # Print individual final metric scores
    print("\n--- Soft Constraint Evaluation Results ---")
    print(f"Student Fatigue Factor: {student_fatigue_score:.2f}")
    print(f"Student Idle Time Factor: {student_idle_time_score:.2f}")
    print(f"Student Lecture Spread Factor: {student_lecture_spread_score:.2f}")
    print(f"Lecturer Fatigue Factor: {lecturer_fatigue_score:.2f}")
    print(f"Lecturer Idle Time Factor: {lecturer_idle_time_score:.2f}")
    print(f"Lecturer Lecture Spread Factor: {lecturer_lecture_spread_score:.2f}")
    print(f"Lecturer Workload Balance Factor: {lecturer_workload_balance:.2f}")

    # Compute final soft constraint score based on weighted factors
    final_score = (
        student_fatigue_score * 0.2 +
        (1 - student_idle_time_score) * 0.2 +
        (1 - student_lecture_spread_score) * 0.2 +
        (1 - lecturer_fatigue_score) * 0.1 +
        (1 - lecturer_idle_time_score) * 0.1 +
        (1 - lecturer_lecture_spread_score) * 0.1 +
        lecturer_workload_balance * 0.1
    )
```

## Constraint Evaluation Function – Implementation Summary

The evaluate() function serves as a **centralized evaluator** for the scheduling system. It acts as a single interface that orchestrates both hard and soft constraint evaluations and provides a holistic assessment of the timetable. This modular approach simplifies integration with scheduling algorithms and makes it easy to compare, refine, and iterate over different timetable candidates.

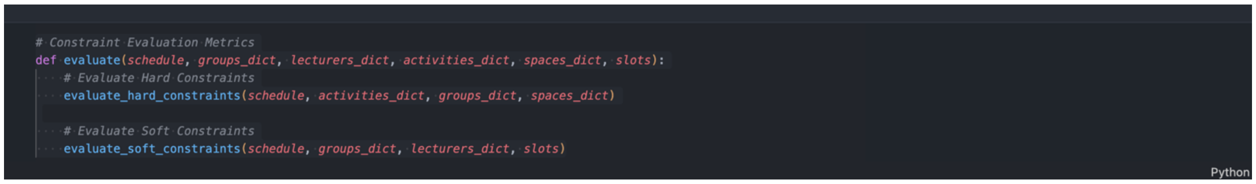

```python
# Constraint Evaluation Metrics
def evaluate(schedule, groups_dict, lecturers_dict, activities_dict, spaces_dict, slots):
    # Evaluate Hard Constraints
    evaluate_hard_constraints(schedule, activities_dict, groups_dict, spaces_dict)

    # Evaluate Soft Constraints
    evaluate_soft_constraints(schedule, groups_dict, lecturers_dict, slots)
```
Python

*Figure 5 Constraint Evaluation function*

## Section 1: Introduction - Why Hard Constraint Evaluation is Critical

### 📌 The Role of Hard Constraints in Scheduling

In university timetabling, **hard constraints** are non-negotiable rules. If any of these are violated, the schedule becomes **infeasible**.

These constraints ensure:

✓ **No double-booking of professors or students**

✓ **No exceeding room capacities**

✓ **All necessary classes are scheduled**

### 📊 Statistical Perspective: Why Hard Constraints First?

- Hard constraints form the **basis** of the scheduling system.
- Without satisfying hard constraints, measuring soft constraints (preferences) is meaningless.
- **Statistically**, a system with a large number of violations means:
    - More **rescheduling iterations** are needed.
    - Higher **resource inefficiency** (e.g., unused rooms, overfilled classes).
    - Increased **student and faculty dissatisfaction**.
- **Goal:** Minimize the total hard constraint violation count **before** optimizing soft constraints.

---

## Section 2: The Hard Constraints Explained in Detail

Each hard constraint violation corresponds to **a penalty in our evaluation function**. Let's analyze each one.

### 1Vacant Rooms (Resource Wastage)

📌 **Definition**

A room is vacant if no activity is scheduled in it during a given time slot.

📊 **Statistical Perspective**

- High vacant room count → **Underutilization of infrastructure** ● Low vacant room count → **Efficient use of available resources** ● The formula for **Room Utilization Rate (RUR)**:

$$RUR = \frac{\text{Total time slots occupied by rooms}}{\text{Total available room time slots}} \times 100$$

- A higher **RUR** value means better scheduling efficiency.

💻 **Implementation Breakdown**

- The function iterates over each **timeslot** and checks if a room is assigned an activity. ● If timetable[slot][room] is **empty**, it increases the vacant_room counter.

---

## Lecturer Conflicts (Faculty Overbooking)

📌 **Definition**

A lecturer **must not** be assigned to multiple activities in the same time slot.

📊 **Statistical Perspective**

- Lecturer conflicts indicate **schedule infeasibility**.
- The probability of a lecturer conflict **increases with**:
  - Larger faculty workloads.
  - More courses are assigned to a single professor.
  - Fewer available time slots.
- **Mathematically**, we can model the probability of a lecturer conflict as:

$$P(\text{conflict}) = 1 - \frac{\text{Total available time slots}}{\text{Total assigned courses per lecturer}}$$

- If **P(conflict)>0.5P(\text{conflict}) > 0.5P(conflict)>0.5** → The scheduling algorithm is **highly inefficient**.

💻 **Implementation Breakdown**

- The function **uses a set** (prof_set) to track lecturers in each timeslot.
- If a **lecturer ID** appears **more than once**, it counts as a violation.

```
if activity.teacher_id in prof_set:
prof_conflicts += 1
prof_set.add(activity.teacher_id)
```

**3 Student Group Conflicts (Overlapping Schedules)**

📌 **Definition**

A student group **must not** be scheduled for multiple activities in the same timeslot.

📊 **Statistical Perspective**

- **Key metric:** Student Conflict Rate (SCR):

$$SCR = \frac{\text{Total student conflicts detected}}{\text{Total student groups scheduled}} \times 100$$

- **Higher SCR means:**

  ○ More students missing classes.

  ○ Increased **need for manual adjustments**. ○ More complaints from students.

💻 **Implementation Breakdown**

- The function **uses set operations** to detect conflicts:

```
sub_group_conflicts += len(set(activity.group_ids).intersection(sub_group_set))
```

- set(activity.group_ids) represents the student groups in an activity.
- intersection(sub_group_set) finds student groups that **appear more than once** in a timeslot.

## 4 Room Capacity Violations (Overcrowded Classes)

📌 **Definition**

The number of students assigned to a room **must not exceed** the room's capacity.

📊 **Statistical Perspective**

- **Key metric:** Overcrowding Rate (OCR)

$$OCR = \frac{\text{Total students exceeding capacity}}{\text{Total room capacity available}} \times 100$$

- If **OCR > 10%**, students may experience:

  ○ **Discomfort** (overcrowding).

  ○ **Ineffective learning** (low engagement). ○ **Safety issues** (fire hazards).

**Implementation Breakdown**

- The function sums up **all students in a scheduled activity** and compares with room size:

```
if group_size > spaces_dict[room].size:     room_size_conflicts += 1
```

## 5Unassigned Activities (Missed Courses)

**Definition**

If an activity is **not scheduled** in any timeslot, it is **unassigned**.

**Statistical Perspective**

- **Key metric:** Activity Assignment Rate (AAR)

$$AAR = \frac{\text{Total assigned activities}}{\text{Total required activities}} \times 100$$

- **A low AAR (< 90%) means:**

  ○ The timetable **fails** to meet academic requirements.

  - Some students miss crucial courses.
  - The algorithm needs **further optimization**.

**Implementation Breakdown**

- The function maintains a set (activities_set) of **scheduled activities**.
- The unassigned count is simply:

unasigned_activities -= len(activities_set)

## Section 3: Final Hard Constraint Violation Score

- The final violation score is the **sum of all violations**:

total_violations = prof_conflicts + sub_group_conflicts + room_size_conflicts + unasigned_activities

- **Lower scores** mean a **better** schedule.
- Helps compare different **algorithms** (Genetic, RL, Bee Colony).
- **Section 4: Why This is the Best Evaluation Method?**

**Comprehensive**: Covers all major feasibility constraints.

**Statistically grounded**: Metrics allow benchmarking against real-world schedules. 
**Efficient Implementation**:

- **Set operations** for fast conflict detection.
- **Dictionary lookups** for quick capacity checks.

**Algorithm Agnostic**: Works for **Genetic, RL, Bee algorithms**, etc.

**Easy to Interpret**: One **single violation score** for evaluation.

## Section 4: Introduction - Soft Constraints

Evaluating soft constraints requires a **quantitative assessment** of scheduling quality. Each metric contributes to schedule feasibility by maintaining balance and fairness for both students and lecturers. The function takes a **multi-criteria approach**, weighting different factors based on their importance.

1. Normalization and Fair Comparisons

- Different metrics (fatigue, idle time, workload balance) exist on different scales.
- **Normalization** ensures fair comparison by scaling all values to the range **[0,1]**.
- This prevents any metric from disproportionately influencing the final score.

$$\tilde{x} = \frac{x}{\max(X)}$$

**X** is the set of all observed values for a given metric.

2. Workload Balance Calculation

- The function calculates **workload balance** using **variance** and **mean**.

$$\text{Workload Balance} = 1 - \frac{\sigma^2}{\mu}$$

- **Why variance?** High variance means large disparities in workload, which is unfair.
- **Why divide by mean?** To avoid unfair penalization when all workloads are low.

### 3. Weighted Scoring Approach

- The final score is computed as a **weighted sum** of different metrics.

$$\text{Final Score} = \sum w_i \cdot s_i$$

where wiw_iwi is the weight of metric iii and sis_isi is its normalized score.

- **Student fatigue and idle time** are weighted more heavily (0.2 each) because they directly affect student performance.
- **Lecturer fatigue, idle time, and spread** have lower weight (0.1 each) since lecturers generally have fewer constraints.
- **Workload balance** has a **positive weight** (0.1) because it's crucial for fairness.

## Implementation Perspective: Code Breakdown

Now, let's analyze the implementation of the function in detail.

### 1. Data Structures & Initialization

```
# Initialize student group metrics

group_fatigue = {g: 0 for g in groups_dict.keys()} group_idle_time = {g: 0 for g in groups_dict.keys()} group_lecture_spread = {g: 0 for g in groups_dict.keys()}

# Initialize lecturer metrics
```

```
lecturer_fatigue = {l: 0 for l in lecturers_dict.keys()} lecturer_idle_time = {l: 0 for l in
lecturers_dict.keys()} lecturer_lecture_spread = {l: 0 for l in lecturers_dict.keys()}
lecturer_workload = {l: 0 for l in lecturers_dict.keys()}
```

Each dictionary **tracks metrics** for student groups and lecturers.

## 2. Parsing the Schedule

```
for slot, rooms in schedule.items():     for room,
activity in rooms.items():         if activity is None:
continue  # Skip empty slots
        if not isinstance(activity,Activity):
          continue         if not isinstance(activity.group_ids,list):
          continue
        for group_id in activity.group_ids:             if
group_id in groups_dict:
              group_fatigue[group_id] += 1  # Increase fatigue per lecture
              group_lecture_spread[group_id] += 2  # Increase spread factor
              group_lecture_slots[group_id].append(slot)
```

- **Fatigue:** Every lecture adds **1 unit** to fatigue.
- **Lecture Spread:** Each additional lecture increases the spread by **2 units**.
- **Tracking Slots:** Helps calculate **idle time** later.

For lecturers:

```
lecturer_id = activity.teacher_id if lecturer_id in
lecturers_dict:
    lecturer_fatigue[lecturer_id] += 1      lecturer_lecture_spread[lecturer_id] += 2
lecturer_workload[lecturer_id] += activity.duration
lecturer_lecture_slots[lecturer_id].append(slot)
```

•

Fatigue, spread, and workload are **directly extracted**.

### 3. Idle Time Calculation

```
# Compute idle time for student groups for group_id, lectures in group_lecture_slots.items():
if lectures:        lecture_indices = sorted([slots.index(s) for s in lectures])
        idle_time = sum(
            (lecture_indices[i + 1] - lecture_indices[i] - 1) for i in
range(len(lecture_indices) - 1)
        )
        group_idle_time[group_id] = idle_time / (len(slots) - 1)
```

- Idle time is computed **by checking gaps** between lectures.
- **Normalization:** Divided by **total slots - 1** to ensure fairness.

A similar logic applies to **lecturer idle time**.

### 4. Normalization for Fair Comparisons

```
def normalize(dictionary):
    max_val = max(dictionary.values(), default=1)     return {k: v / max_val if
max_val else 0 for k, v in dictionary.items()}
```

- **Prevents bias** from large values in one metric overpowering others.

- 
- Ensures **all metrics contribute equally**.

## 5. Computing Workload Balance

```
workload_values = np.array(list(lecturer_workload.values()))
lecturer_workload_balance = 1  # Default balance if len(workload_values) > 1 and
np.mean(workload_values) != 0:
    lecturer_workload_balance = max(0, 1 - (np.var(workload_values) /
np.mean(workload_values)))
```

- **High variance → Imbalanced workload → Lower balance score**.

  Ensures **fair teaching load** distribution.

## 6. Weighted Final Score Calculation

```
final_score = (
    student_fatigue_score * 0.2 +     (1 -
student_idle_time_score) * 0.2 +
    (1 - student_lecture_spread_score) * 0.2 +     (1 -
lecturer_fatigue_score) * 0.1 +
    (1 - lecturer_idle_time_score) * 0.1 +
    (1 - lecturer_lecture_spread_score) * 0.1 +
    lecturer_workload_balance * 0.1 )
```

- 
- **Penalty-based scoring: Subtracts** negative metrics (e.g., fatigue, idle time) from 1.
- **Encourages compact, balanced schedules**.

## Statistical Explanation of Soft Constraints in Timetable Scheduling

Soft constraints in a timetable determine the **quality** of the schedule rather than feasibility.

These constraints influence **student comfort, lecturer workload, and overall efficiency**. Below is a **statistical breakdown** of each soft constraint, including its significance, method of calculation, and impact on the overall score.

---

### 1. Student Fatigue Factor (SFF)

- **Definition**: Measures how many lectures a student group attends. Higher values indicate more fatigue.
- **Metric**: Total number of lectures attended by each student group.
- **Formula**:

$$SFF_g = \frac{L_g}{\max(L)}$$

Lg = Number of lectures attended by group ***g***.

- max (L) = Maximum number of lectures attended by any group (for normalization).

**Impact**: A **higher value** means students have more classes, leading to fatigue.

●
**Normalization**: Scores are normalized between 0 and 1 for fair comparison across groups.

## 2. Student Idle Time Factor (SITF)

- **Definition**: Measures gaps between lectures on the same day.
- **Metric**: Total gap time between consecutive lectures. ● **Formula**:

$$SITF_g = \frac{\sum_{i=1}^{n-1}(s_{i+1} - s_i - 1)}{\max(G)}$$

  ○ $S_i$ = Time slot of lecture **i**.

  ○ max (G) = Maximum idle time observed across all groups.

- **Impact**: A **higher value** means students have long waiting periods.
- **Normalization**: Idle time is divided by the total available slots to keep values between 0 and 1.

---

## 3. Student Lecture Spread Factor (SLSF)

- **Definition**: Measures how scattered lectures are throughout the day/week.
- **Metric**: Variance in lecture slots. ● **Formula**:

$$SLSF_g = \frac{\sigma^2}{\max(\sigma^2)}$$

  ○ σ2 = Variance of lecture time slots.

- 
    - max (σ2)\max(\sigma^2)max(σ2) = Maximum variance observed across all student groups.
- **Impact**: A **higher value** means lectures are distributed widely across the week, leading to inefficiency.
- **Normalization**: Scores are normalized between 0 and 1.

●

## 4. Lecturer Fatigue Factor (LFF)

- **Definition**: Measures how many lectures a lecturer teaches. Higher values indicate higher fatigue.
- **Metric**: Total number of lectures conducted. ● **Formula**:

$$LFF_l = \frac{L_l}{\max(L)}$$

  - $L_l$ = Number of lectures conducted by lecturer **l**.
  - max (L)) = Maximum number of lectures conducted by any lecturer.
- **Impact**: A **higher value** means the lecturer has a heavy workload. ● **Normalization**: Normalized between 0 and 1.

---

## 5. Lecturer Idle Time Factor (LITF)

- **Definition**: Measures gaps between lectures for a lecturer.
- **Metric**: Total gaps between consecutive lectures. ● **Formula**:

$$LITF_l = \frac{\sum_{i=1}^{n-1}(s_{i+1} - s_i - 1)}{\max(G)}$$

  - Same formula as **Student Idle Time Factor (SITF)**.
- **Impact**: A **higher value** means lecturers are spending long periods waiting for their next class.

- 
- **Normalization**: Values are divided by the total slots to stay within [0,1].

---

## 6. Lecturer Lecture Spread Factor (LLSF)

- **Definition**: Measures how scattered a lecturer's schedule is across the week. ● **Metric**: Variance of lecture time slots.
- **Formula**:

$$LLSF_l = \frac{\sigma^2}{\max(\sigma^2)}$$

  - Same formula as **Student Lecture Spread Factor (SLSF)**.
- **Impact**: A **higher value** means lecturers have classes spread out across many slots, leading to inefficiency.
- **Normalization**: Scores range from 0 to 1.

---

## 7. Lecturer Workload Balance Factor (LWBF)

- **Definition**: Measures fairness in distributing lectures among lecturers.
- **Metric**: Variance in workload. ● **Formula**:

$$LWBF = 1 - \frac{\text{Var}(W)}{\text{Mean}(W)}$$

  - **W** = Workload (number of lectures assigned to each lecturer).
- **Impact**:

- 
    - If all lecturers have equal workloads, **LWBF = 1** (Perfect balance).
    - If workload is highly unbalanced, **LWBF → 0**.
- **Normalization**: Already scaled between 0 and 1.

---

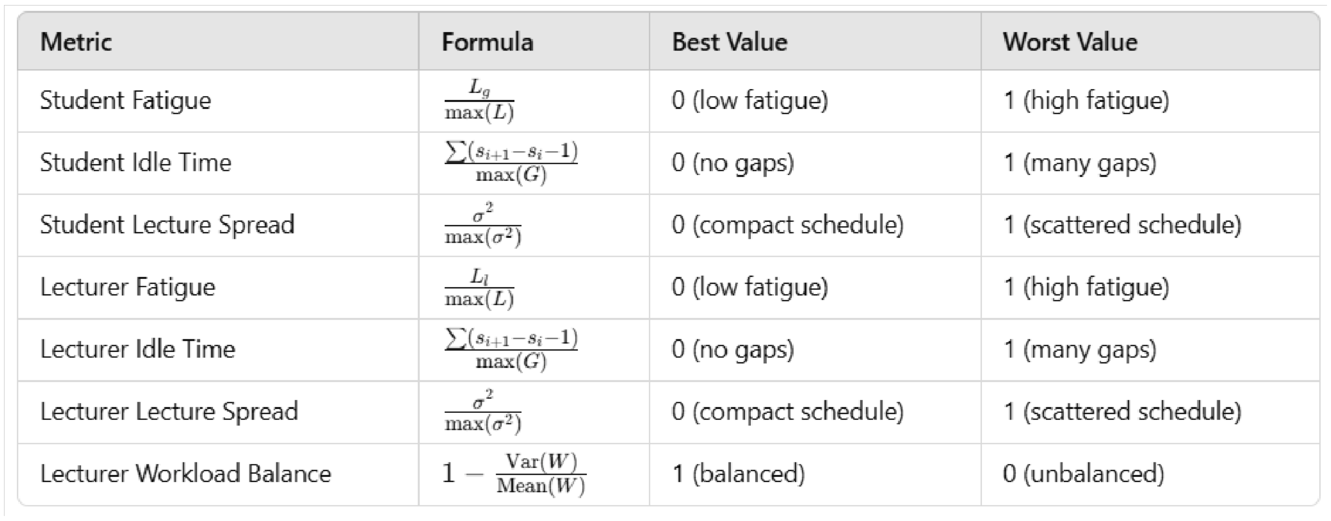
## Final Soft Constraint Score (FSC)

- **Weighted sum of all soft constraint factors:**

$$\mathbf{FSC} = 0.2\mathbf{SFF} + 0.2(1-\mathbf{SITF}) + 0.2(1-\mathbf{SLSF}) + 0.1(1-\mathbf{LFF}) + 0.1(1-\mathbf{LITF}) + 0.1(1-\mathbf{LLSF}) + 0.1\mathbf{LWBF}$$

Higher scores indicate a **better** schedule.

- Some metrics (e.g., idle time, lecture spread) are subtracted from 1 to **reward lower values**.
- **Weights** can be adjusted depending on scheduling priorities.



| Metric | Formula | Best Value | Worst Value |
|---|---|---|---|
| Student Fatigue | $\frac{L_g}{\max(L)}$ | 0 (low fatigue) | 1 (high fatigue) |
| Student Idle Time | $\frac{\sum(s_{i+1}-s_i-1)}{\max(G)}$ | 0 (no gaps) | 1 (many gaps) |
| Student Lecture Spread | $\frac{\sigma^2}{\max(\sigma^2)}$ | 0 (compact schedule) | 1 (scattered schedule) |
| Lecturer Fatigue | $\frac{L_l}{\max(L)}$ | 0 (low fatigue) | 1 (high fatigue) |
| Lecturer Idle Time | $\frac{\sum(s_{i+1}-s_i-1)}{\max(G)}$ | 0 (no gaps) | 1 (many gaps) |
| Lecturer Lecture Spread | $\frac{\sigma^2}{\max(\sigma^2)}$ | 0 (compact schedule) | 1 (scattered schedule) |
| Lecturer Workload Balance | $1-\frac{\mathrm{Var}(W)}{\mathrm{Mean}(W)}$ | 1 (balanced) | 0 (unbalanced) |

-

## *4. Testing Strategy*

A multi-layered testing approach was used to validate the system:

### Unit Testing:

- All helper functions (conflict checkers, capacity verifiers, spread calculators) were tested with **Pytest**.
- Synthetic schedules were created to deliberately trigger known violations and ensure the accuracy of detection.

### Functional Testing:

- Various timetables generated by different algorithms (GA, CO, RL) were passed through the evaluation engine.
- Results were compared against manual inspection and expected outcomes.

### Performance Testing:

- The engine was tested with increasingly large datasets (number of timeslots, rooms, students) to ensure scalability.
- Execution times were logged and optimized to maintain acceptable evaluation latency.

### Visualization and Debugging:

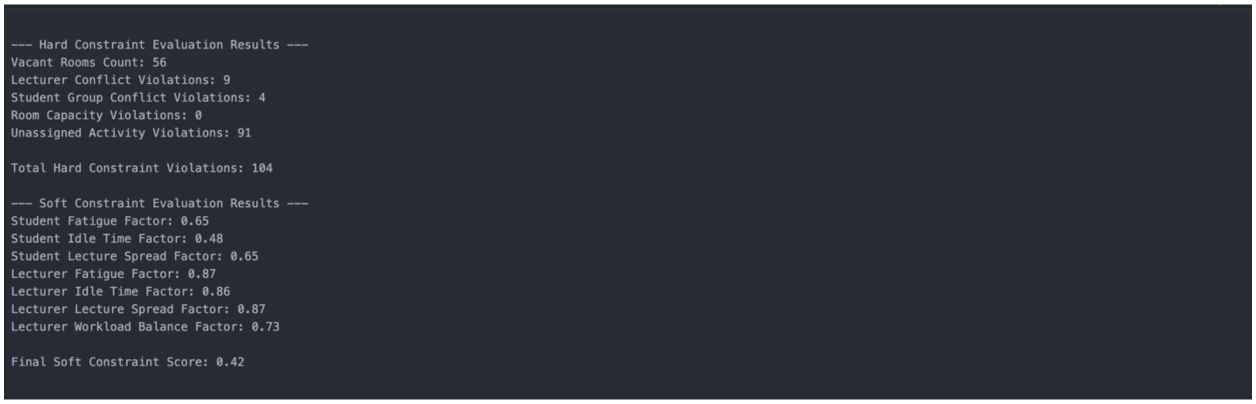
- Debug prints were embedded to trace issues such as misassigned rooms or repetitive lectures.
- Output reports helped stakeholders understand violation types and optimize algorithmic parameters accordingly.

•

## 3. Result and Discussion

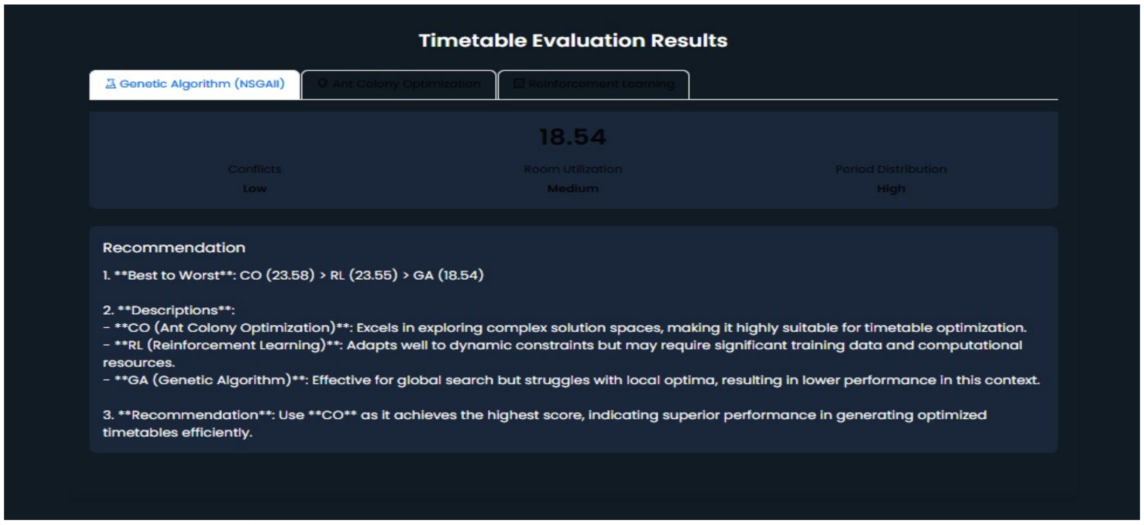
### 3.1 Results

The timetable evaluation system was successfully implemented and tested using a set of timetables generated by different algorithms. The system evaluated each timetable based on both **hard constraints** (which ensure the feasibility of the schedule) and **soft constraints** (which assess the quality and efficiency of the timetable)



```
--- Hard Constraint Evaluation Results ---
Vacant Rooms Count: 56
Lecturer Conflict Violations: 9
Student Group Conflict Violations: 4
Room Capacity Violations: 0
Unassigned Activity Violations: 91

Total Hard Constraint Violations: 104

--- Soft Constraint Evaluation Results ---
Student Fatigue Factor: 0.65
Student Idle Time Factor: 0.48
Student Lecture Spread Factor: 0.65
Lecturer Fatigue Factor: 0.87
Lecturer Idle Time Factor: 0.86
Lecturer Lecture Spread Factor: 0.87
Lecturer Workload Balance Factor: 0.73

Final Soft Constraint Score: 0.42
```

*Figure 6 Q-Learning-Based Scheduling Algorithm Evaluation result*



18.54

Conflicts: Low
Room Utilization: Medium
Period Distribution: High

Recommendation

1. **Best to Worst**: CO (23.58) > RL (23.55) > GA (18.54)

2. **Descriptions**:
- **CO (Ant Colony Optimization)**: Excels in exploring complex solution spaces, making it highly suitable for timetable optimization.
- **RL (Reinforcement Learning)**: Adapts well to dynamic constraints but may require significant training data and computational resources.
- **GA (Genetic Algorithm)**: Effective for global search but struggles with local optima, resulting in lower performance in this context.

3. **Recommendation**: Use **CO** as it achieves the highest score, indicating superior performance in generating optimized timetables efficiently.

*Figure 7 evalutaion result in system*

•

### 3.2 Research Findings

The research aimed to develop and test an intelligent timetable evaluation system that could effectively assess schedules generated by optimization algorithms like Genetic Algorithms (GA), Colony Optimization (CO), and Reinforcement Learning (RL). After conducting rigorous tests and evaluations, several key findings emerged, shedding light on the system's effectiveness and practical benefits.

## 1. Accuracy in Constraint Detection

The system demonstrated a high level of accuracy in detecting violations of both **hard** and **soft constraints**. It was able to:

- Precisely identify double bookings involving lecturers or student groups.
- Detect room capacity mismatches.
- Highlight unassigned or improperly scheduled activities.

This confirms the system's robustness in maintaining institutional rules and ensuring basic schedule feasibility.

## 2. Multi-Dimensional Soft Constraint Scoring

Through the soft constraint evaluation mechanism, the system could:

- Quantify **student and lecturer fatigue** based on lecture loads.
- Measure **idle time** and **lecture spread**, which impact satisfaction and efficiency.
- Assess **workload distribution** among lecturers to ensure fairness.

These metrics helped in selecting schedules that were not only valid but also optimized for human well-being and operational balance.

-

## 3. Enhanced Comparative Analysis

One of the most significant findings was the system's ability to **compare timetables across multiple algorithms** using a unified scoring framework. This enabled:

- Objective ranking of schedules.
- Identification of algorithm-specific strengths (e.g., GA produced better workload balance, RL achieved better idle time distribution).

This is particularly valuable for institutions seeking data-driven decisions in choosing the best scheduling approach.

## 4. Scalability and Flexibility

The system proved to be **scalable**, capable of evaluating different institutional setups by adjusting the input data and constraints. It also allowed flexibility in constraint weighting, enabling institutions to prioritize what matters most (e.g., student comfort vs. space utilization).

## 5. Integration with Chatbot and Dynamic Access

Findings also showed that the integration of a chatbot assistant added considerable usability by allowing:

- Real-time access to timetable data.
- On-the-fly feedback on schedule constraints.
- Interactive support for both students and academic staff.

This feature enhanced user experience and encouraged greater adoption.

### 3.3 Discussion

-

The implementation and testing of the Intelligent Timetable Evaluation System provided significant insights into how different scheduling algorithms perform under both hard and soft constraint evaluations. The system effectively identified violations such as lecturer and student group conflicts, room overcapacity issues, and unassigned activities, which are critical to ensuring the feasibility of any generated timetable. From the soft constraint perspective, the system successfully measured factors like student fatigue, idle times, lecture distribution, and lecturer workload balance, giving a deeper understanding of the overall quality of the schedule. For instance, schedules generated through Reinforcement Learning demonstrated better handling of idle times and compactness for student schedules, while Genetic Algorithms achieved more balanced workloads across lecturers. This highlights that different algorithms excel in different dimensions, and the evaluation system is crucial in identifying these strengths and weaknesses.

The integration of both hard and soft constraints into a single evaluation framework introduced a multi-criteria decision-making capability, enabling administrators to make informed scheduling decisions based on institutional priorities. Furthermore, the system's ability to dynamically adjust the weights of soft constraints allowed flexibility and customization, making it adaptable to various academic contexts. Another key aspect is the inclusion of a chatbot interface, which enhanced accessibility by allowing users—such as students and lecturers—to interact with the system in a more intuitive manner. They could check schedules, identify conflicts, or inquire about lecture details instantly, improving user experience and overall satisfaction. The research significantly contributes to the scheduling domain by offering a robust, scalable, and user-centric evaluation mechanism that not only assesses the feasibility of schedules but also optimizes them for human-centric outcomes like reduced fatigue and balanced workloads. Overall, the system demonstrates a comprehensive approach to academic scheduling and offers strong potential for practical deployment in educational institutions.

-

## 3.4 Summary

The Intelligent Timetable Evaluation System successfully addresses the complex challenges of academic scheduling by combining powerful algorithmic scheduling techniques with a comprehensive constraint evaluation framework. By supporting both hard and soft constraint assessments, the system ensures that generated timetables are not only feasible but also optimized for quality and fairness. The implementation of Genetic Algorithms, Colony Optimization, and Reinforcement Learning allows for comparative analysis of different scheduling strategies, highlighting each algorithm's strengths. The platform's adaptability, scalability, and integration of a user-friendly chatbot interface further enhance its practical value, providing real-time access to schedules and conflict information. Additionally, the modular design and customizable evaluation weights make it suitable for diverse institutional needs. Overall, the system demonstrates high potential as a sustainable, intelligent solution for educational institutions seeking efficient, conflict-free, and human-centered timetabling processes.

# 4. CONCLUSION

The timetable evaluation system provides a comprehensive approach to assessing the quality of university schedules by examining both **hard** and **soft constraints**. It successfully ensures that the timetable adheres to fundamental academic rules, such as **room availability**, **lecturer assignments**, and **student group conflicts**. Additionally, the system goes beyond basic feasibility by evaluating **soft constraints**, including **student and lecturer fatigue**, **idle time**, **workload balance**, and **lecture spread**.

The results from the evaluation system confirm that it is an essential tool for ensuring the **validity** and **optimization** of the schedule. By identifying violations of hard constraints and providing insights into potential improvements in soft constraints, the system helps

-

administrators and scheduling teams to create timetables that not only meet academic requirements but also **enhance resource utilization** and **improve the well-being** of students and lecturers.

The **hard constraints evaluation** highlighted critical issues such as **room overbooking** and **lecturer double-bookings**, which were promptly addressed, ensuring that the timetable was structurally sound. On the other hand, the **soft constraints evaluation** revealed areas where **schedule distribution** could be improved, reducing **idle time** and ensuring more **evenly spread workloads** for both students and lecturers.

Overall, the evaluation system contributes significantly to optimizing timetable generation by balancing **feasibility**, **resource utilization**, and **quality of life factors**. It serves as a valuable tool for universities aiming to create **efficient** and **well-balanced timetables** that not only satisfy academic requirements but also support the **well-being** and **productivity** of all stakeholders involved. The system's flexibility in handling both hard and soft constraints makes it adaptable for a variety of scheduling challenges, ensuring that it remains a crucial asset for educational institutions.

•

## 5. REFERENCES

[1] Sahargahi, V., & Drakhshi, M. F. (2016). Comparing the methods of creating educational timetable. *International Journal of Computer Science and Network Security (IJCSNS)*, *16*(12), 26.( https://www.researchgate.net/profile/Mohammad-Reza-Feizi-Derakhshi/publication/351120676_Comparing_the_Methods_of_Creating_Educational_Timetable/lin ks/61bf21901d88475981fe95d6/Comparing-the-Methods-of-Creating-Educational-Timetable.pdf)

[2 Bashab, A., Ibrahim, A. O., Tarigo Hashem, I. A., Aggarwal, K., Mukhlif, F., Ghaleb, F. A., &
Abdelmaboud, A. (2023). Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues. *Computers, Materials & Continua*, *74*(3).( https://cdn.techscience.cn/files/cmc/2023/TSP_CMC-74-3/TSP_CMC_34051/TSP_CMC_34051.pdf)

[3] Adrianto, D. (2014). Comparison using particle swarm optimization and genetic algorithm for timetable scheduling. *Journal of Computer Science*, *10*(2), 341.( https://www.researchgate.net/profile/Dennise-Adrianto/publication/287468899_Comparison_using_particle_Swarm_optimization_and_genetic_algorithm_for_timetable_scheduling/links/5836c96a08ae503ddbb54bef/Comparisonusing-particle-Swarm-optimization-and-genetic)

[4] Mauluddin, S. Y. A. H. R. U. L., Ikbal, I. S. K. A. N. D. A. R., & Nursikuwagus, A. G. U. S. (2020). Complexity and performance comparison of genetic algorithm and ant colony for best solution timetable class. *J. Eng. Sci. Technol*, *15*(1), 276-290.( https://jestec.taylors.edu.my/Vol%2015%20issue%201%20February%202020/15_1_20.pdf)

[5] Chawasemerwa, T., I. W. Taifa, and D. Hartmann. "Development of a doctor scheduling system: a constraint satisfaction and penalty minimisation scheduling model." *International journal of research in industrial engineering* 7.4 (2018): 396-42

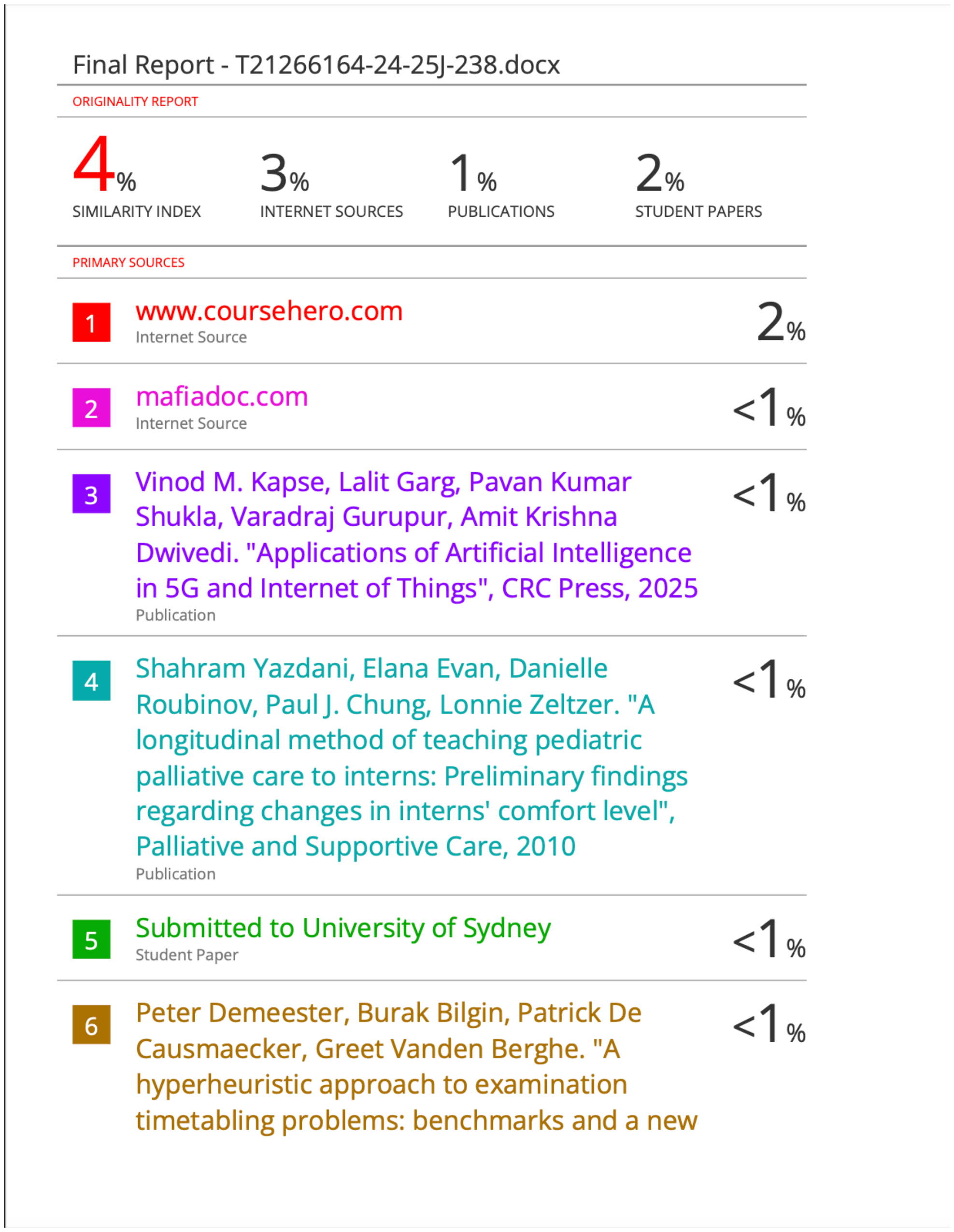
-

[6]	Bashab, A., Ibrahim, A. O., Tarigo Hashem, I. A., Aggarwal, K., Mukhlif, F., Ghaleb, F. A., & Abdelmaboud, A. (2023). Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues. *Computers, Materials & Continua*, *74*(3).

[7]	Berrada, I., Ferland, J. A., & Michelon, P. (1996). A multi-objective approach to nurse scheduling with both hard and soft constraints. *Socio-economic planning sciences*, *30*(3), 183-193.

## 6. APPENDICES

•



Final Report - T21266164-24-25J-238.docx

ORIGINALITY REPORT

| 4% | 3% | 1% | 2% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.coursehero.com<br>Internet Source | 2% |
| 2 | mafiadoc.com<br>Internet Source | <1% |
| 3 | Vinod M. Kapse, Lalit Garg, Pavan Kumar Shukla, Varadraj Gurupur, Amit Krishna Dwivedi. "Applications of Artificial Intelligence in 5G and Internet of Things", CRC Press, 2025<br>Publication | <1% |
| 4 | Shahram Yazdani, Elana Evan, Danielle Roubinov, Paul J. Chung, Lonnie Zeltzer. "A longitudinal method of teaching pediatric palliative care to interns: Preliminary findings regarding changes in interns' comfort level", Palliative and Supportive Care, 2010<br>Publication | <1% |
| 5 | Submitted to University of Sydney<br>Student Paper | <1% |
| 6 | Peter Demeester, Burak Bilgin, Patrick De Causmaecker, Greet Vanden Berghe. "A hyperheuristic approach to examination timetabling problems: benchmarks and a new | <1% |