Advanced Timetable Generation Solution for Educational Institutes

Project ID- 24-25J-238

De Silva K.H.P.N-IT21208980

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

Department of Information Technology

Sri Lanka Institute of Information Technology Sri Lanka

July 2024

Reinforcement Learning Algorithm Using Advanced Timetable Generation Solution for Educational Institutes

Project ID- 24-25J-238

De Silva K.H.P.N-IT21208980

Supervisor : Mr.Jeewka Perara

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

Department of Information Technology

Sri Lanka Institute of Information Technology Sri Lanka

July 2024

DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
De Silva K.H.P.N	IT21208980	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

ABSTRACT

Planning timetables in instructive teaching presents significant challenges due to the complex administration of limitations including resources, accessibility, and client preferences. This report presents an imaginative strategy that blends Dynamic Multi-Objective Evolutionary Problems (DMOEPs) with Multi-Criteria Decision Making (MCDM) to improve the optimization of timetables. DMOEPs are utilized to handle different, regularly clashing targets, such as minimizing hold-up times and maximizing resource utilization, whereas MCDM provides an organized system to prioritize and adjust these destinations based on user-defined criteria. This coordination approach guarantees that the framework can adjust to both built up client imperatives and startling natural changes, such as sudden workforce absences or issues with room accessibility. Test findings demonstrate that the combination of DMOEPs and MCDM provides a strong and flexible arrangement for complex planning challenges, particularly progressing planning quality and versatility in comparison to conventional strategies.

Keywords: Timetable Scheduling, Dynamic Multi-Objective Evolutionary Problems, MultiCriteria Decision Making, Dynamic Optimization, Educational Institutions, Resource Allocation,

Constraint Management

ACKNOWLEDGEMENT

I would like to begin by expressing my sincere gratitude to my professors and advisers, whose guidance and support have been invaluable throughout the development of this project proposal. Their expertise, patience, and constructive criticism have been instrumental in shaping the direction and success of this proposal. The time and effort they invested in mentoring me has greatly enhanced this work's quality, and I am deeply thankful for that. I am also grateful to my colleagues and peers, whose constructive discussions and suggestions helped refine my ideas and approach. I greatly appreciate their collaborative spirit and willingness to share knowledge. Special thanks to the universities, institutes, and libraries for providing the necessary resources and a conducive environment for research. Without their cooperation, it would have been impossible to gather the data and insights needed to formulate this proposal. I would also like to acknowledge the support and understanding of my family and friends, who stood by me during the challenging times and provided the moral support needed to complete this work. It is with great appreciation that I recognize the many individuals and institutions whose efforts and contributions have made the completion of this proposal possible. Thank you all for your invaluable assistance and support. Lastly, I thank everyone who, directly or indirectly, contributed to this proposal. Your encouragement and assistance have made this journey a fulfilling and enriching experience.

CONTEST

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
CSV	Comma-Separated Values
DQL	Deep Q-Learning
MCDM	Multi-Criteria Decision Making
AHP	Analytic Hierarchy Process
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
RL	Reinforcement Learning
DMOEPs	Dynamic Multi-Objective Evolutionary Problems
ETL	Extract, Transform, Load
PSO	Particle Swarm Optimization
GA	Genetic Algorithm

1.Introduction

Timetable scheduling in educational institutions could be a multifaceted issue characterized by complex optimization and compliance challenges. Conventional planning strategies frequently fall short regarding the energetic nature of these issues, as they battle to adjust to real-time changes in limitations and necessities. To address these challenges, we propose a progressed approach that combines Multi-Criteria Decision-Making (MCDM) with procedures for tackling Dynamic Multi-Objective Evolutionary Problems (DMOEPs) [1] [2] [3].

The essential challenge in instructive timetable planning is overseeing and optimizing a huge number of limitations and destinations, such as staff accessibility, room capacity, and course prerequisites. Conventional strategies, such as counting hereditary calculations and heuristic approaches, are regularly constrained by their failure to powerfully alter real-time changes and clashing targets [4]. Our approach involves coordinating MCDM methods to efficiently assess and prioritize numerous criteria, guaranteeing an adjusted optimization of planning results. By consolidating MCDM, we address the different measurements of the planning issue, such as tradeoffs between diverse planning targets and limitations [5].

In addition to MCDM, we utilize procedures from DMOEPs to handle the energetic nature of planning issues. DMOEPs are designed to track and adjust to changes within the optimization scene, making them well-suited for advancing planning challenges where targets and constraints can vary over time [1]. Our approach utilizes support learning-based instruments to successfully react to changes within the planning environment, such as unforeseen staff nonattendances or fluctuating understudy enrollment designs.

This double integration of MCDM and DMOEP procedures gives a vigorous system for powerfully optimizing timetables, enhancing adaptability and versatility compared to conventional strategies. The viability of this approach is assessed through broad tests, illustrating changes in planning quality and operational productivity.

By combining MCDM with DMOEP procedures, our inquiry offers a comprehensive solution to the complexities of timetable planning in instructive education, setting a modern standard for versatile and effective planning.

1.1.Background and literature survey

In recent years, combining Multi-Criteria Decision Making (MCDM) and optimization methods has gained traction, particularly in complex and energetic situations. MCDM gives an organized approach for assessing and prioritizing different clashing goals, making it a profitable instrument for handling complex planning issues [1] [2].

Educational timetable planning has customarily depended on strategies such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). These strategies, whereas valuable, regularly experience restrictions due to the energetic and multi-objective nature of planning issues. Genetic Algorithms, propelled by normal determination, are proficient at investigating broad arrangement spaces but may endure untimely meetings when managing complex limitations [3]. So also, PSO, which is based on the social behavior of winged creatures or angles, accomplishes fast joining but may not completely investigate the arrangement space in quickly changing situations [4].

Later progressions in optimization procedures have highlighted the benefits of hybrid approaches. For instance, combining Fortification Learning with Developmental Calculations has appeared to guarantee overcoming the restrictions of conventional strategies. These hybrid calculations consolidate versatile learning capabilities, permitting them to adapt to changing circumstances and provide more adaptable arrangements [5].

Energetic Multi-Objective Developmental Issues (DMOEPs) are especially pertinent to the field of instructive planning. DMOEPs are outlined to handle optimization issues with changing destinations and imperatives, making them reasonable for energetic planning situations where conditions such as course sizes or workforce accessibility can vary [4]. Joining DMOEP strategies with MCDM permits the advancement of strong planning calculations that can viably oversee different goals and adjust to energetic conditions.

By speaking to planning issues through a system that joins both MCDM and DMOEP strategies, our approach points to addressing the deficiencies of conventional strategies. This integration guarantees that planning calculations can powerfully alter to real-time changes, advertising more productive and adaptable arrangements compared to ordinary approaches [5].

In conclusion, the combination of MCDM and DMOEP methods speaks to a significant progression in instructive timetable arranging. These techniques provide a solid system for overseeing the complexities of energetic planning situations, guaranteeing that timetables are both optimized and versatile to changing circumstances.

1.2.Research Gap

The application of Reinforcement Learning (RL) in educational scheduling has highlighted a few zones that warrant examination. Traditional timetable optimization strategies, which regularly depend on authentic information, battle to adjust to real-time energetic imperatives and cannot immediately alter plans [1] [2]. Whereas coordination Multi-Criteria Decision Making (MCDM) with RL has appeared to guarantee to tend to these challenges, there's still a significant hole in successfully overseeing the energetic and clashing destinations that emerge in real-time scheduling situations.

Even though hybrid approaches combining RL with Evolutionary Algorithms have illustrated the potential for overcoming the restrictions of traditional strategies, there's a need for more inquiry about the viability of these combinations in energetic planning contexts [3]. Particularly, the capacity of these strategies to adjust to unforeseen changes, such as fluctuating student enrollments or sudden staff unavailability, remains underexplored.

Moreover, current RL-based planning strategies need personalization and versatility centered on client inclinations. Whereas existing methods center on moving forward plans based on predefined criteria, they frequently fall short of accounting for the needs of understudies and staff individuals. Future investigations ought to point to creating RL calculations that consolidate real-time struggle location and personalized alterations, subsequently improving both planning proficiency and client fulfillment [4] [5].

In addition, there's a pressing need for comprehensive benchmarking and comparative consideration of RL-based optimization strategies in dynamic and multi-objective scheduling scenarios. Setting up standardized benchmarks and execution measurements will give profitable bits of knowledge into the qualities and shortcomings of different RL approaches, directing advanced progress in planning solutions [5]. Tending to these crevices will lead to the advancement of more productive, adaptable, and user-friendly scheduling systems

Research | Research Gap	**Dynamic as constraints inputs**	**Multi-Criteria Decision Making**	**Real-Time Optimization**
Deep-Reinforcement Learning based dynamic optimization of bus timetable [1]	NO	NO	YES
A reinforcement learning approach for dynamic multi-objective optimization [3]	YES	NO	YES
A deep reinforcement learning based multi-criteria decision support system for optimizing textile chemical process [5]	NO	YES	YES
A Review of Reinforcement Learning Based Intelligent Optimization for Manufacturing Scheduling [2]	YES	NO	YES
Proposed System	YES	YES	YES

Table 1 - Research Gap

1.3.Research Problem

Reinforcement Learning (RL) shows great potential in enhancing scheduling systems by adjusting to changing conditions and enhancing decision-making processes. Even though it has potential, there are still numerous important research gaps when using RL for educational scheduling.

Dynamic Constraint Handling: Reinforcement learning techniques must improve their capacity to handle changing conditions, such as unexpected classroom schedule changes, teacher missing days, or evolving student requirements. Current methods frequently face challenges in making immediate changes in real-time, resulting in less-than-ideal scheduling outcomes.

Scalability: With the increase in size and complexity of educational organizations, the ability to scale RL-based scheduling systems presents a major obstacle. It is crucial to tackle scalability issues to effectively handle extensive and intricate scheduling problems for wider use.

Integration with Hybrid Approaches: Integrating RL with evolutionary algorithms, simulated annealing, or metaheuristics may enhance both solution quality and efficiency. Examining these hybrid methods could utilize the advantages of various techniques to improve overall efficiency.

User Interaction and Flexibility: It is essential to create user-friendly interfaces that enable educational administrators to interact with and manually modify schedules produced by RL systems. Improving customization and adaptability in scheduling options will more effectively address the varied requirements of students and educators.

Comprehensive Benchmarking: Thorough benchmarking and comparison of RL-based scheduling methods with other optimization techniques are necessary to gain a deeper understanding of their effectiveness. It is crucial to perform thorough comparative research to confirm and improve RL techniques.

1.4. Objectives

1.4.1. Main Objectives

Develop a specialized scheduling optimization system for educational environments using Reinforcement Learning (RL) methods. This system needs to successfully manage changing limitations and offer flexible scheduling options of high quality.

1.4.2. Specific Objectives

1. Design and Develop RL Algorithms for Scheduling:
   - Research and apply RL algorithms suitable for task scheduling, focusing on methods such as Q-Learning, Deep Q-Learning (DQL), and SARSA. Adapt these algorithms to handle intricate scheduling constraints and dynamic, real-time environments.
   - Incorporate Multi-Criteria Decision Making (MCDM) techniques to address conflicting objectives and improve decision-making in the scheduling process.
2. Integrate ETL System for Data Handling:
   - Develop and implement an ETL system to manage and extract data from various user inputs, such as CSV, system, and Excel files.
   - Ensure that the ETL processes seamlessly integrate with the RL-based scheduling optimization system, enabling effective management of diverse data sources.
3. Optimize RL Algorithms for Real-Time Applications:
   - Enhance reinforcement learning models to ensure their effectiveness in real-time applications. Address challenges related to rapid adaptation and optimization using up-to-date data inputs, ensuring the scheduling system remains timely and accurate.
   - Integrate MCDM techniques to refine the RL algorithms, enabling them to handle multiple conflicting objectives and prioritize decision-making criteria in real-time.

4. Implement and Integrate an Interactive User Interface:
    - Develop a user-friendly interface for educational administrators to interact with and manually adjust optimized schedules. Ensure the interface supports real-time modifications while adhering to user-defined constraints and optimization goals.
    - Incorporate MCDM elements into the interface to allow users to easily prioritize and adjust multiple objectives based on their specific needs.
5. Validate and Benchmark RL-Based Scheduling Solutions:
    - Conduct thorough testing and benchmarking of the RL-powered scheduling system against traditional and hybrid optimization techniques. Evaluate the system's performance, scalability, and effectiveness in meeting scheduling requirements, and make improvements as needed.
    - Include assessments that specifically compare the impact of integrating MCDM techniques with RL algorithms on scheduling quality and user satisfaction.

2.Methodology

1. Getting User-Defined Constraints and the Schedule

- Define the timetable optimization problem based on the constraints and objectives provided by the client.
- Gather input constraints and timetable data through a user-friendly interface.
- Implement an ETL (Extract, Transform, Load) system to accommodate and extract needed information from various user inputs (CSV, system, Excel formats).

2. Application of RL Agent

- Implement a single RL agent, such as Q-Learning, Deep Q-Learning (DQL), or SARSA, depending on the complexity and requirements of the problem.
- Design and implement reward mechanisms/functions tailored to scheduling outcomes. Convert user-defined constraints into a reward function, including a weighted sum of each constraint.

3. Integration of Multi-Criteria Decision Making (MCDM)

- Apply MCDM techniques to evaluate and prioritize the different objectives and constraints. Use methods such as Analytic Hierarchy Process (AHP) or Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) to rank and select the best possible scheduling solutions.
- Integrate the MCDM outcomes into the RL agent's decision-making process, ensuring that the selected schedule aligns with the prioritized criteria and objectives.

4. Handling Dynamic Multi-Objective Evolutionary Problems (DMOEPs)

- Adapt the RL agent to handle DMOEPs by incorporating mechanisms that allow the agent to adjust to changes in objectives and constraints over time. This might include dynamically adjusting the reward functions or exploring different evolutionary strategies to optimize multiple conflicting objectives simultaneously.

- Use DMOEP techniques to manage the dynamic aspects of the scheduling problem, such as changes in faculty availability or fluctuating student enrollment.

5. Optimization Process

- The RL agent interacts with the timetable model to explore and exploit potential solutions, guided by the MCDM-ranked criteria and DMOEP strategies.
- Continuously evaluate and fine-tune the performance of the agent in meeting the defined dynamic constraints and objectives.
-

6. Selection of the Optimal Solution

- Analyze the outcomes of the RL agent's learning process, using the results from MCDM and DMOEPs to select the optimal solution that best meets the client's requirements and dynamic constraints.

7. Implementation and Integration

- Integrate the chosen RL model with the user interface.
- Allow for manual adjustments and updates in real-time based on the optimized timetable.

8. Validation and Testing

- Test the optimized timetable through simulations and real-world scenarios.
- Validate the solution against the original constraints and dynamic requirements to ensure effectiveness.

2.1.Overall System Diagram

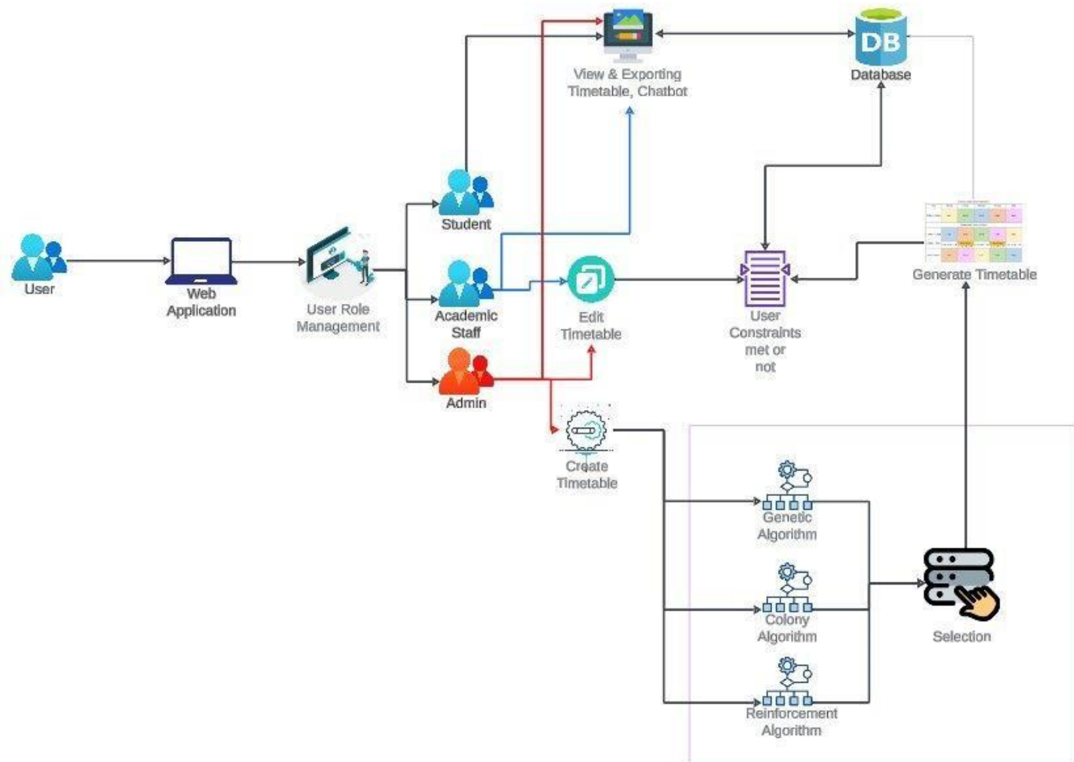


Figure 1 - System Overview Diagram

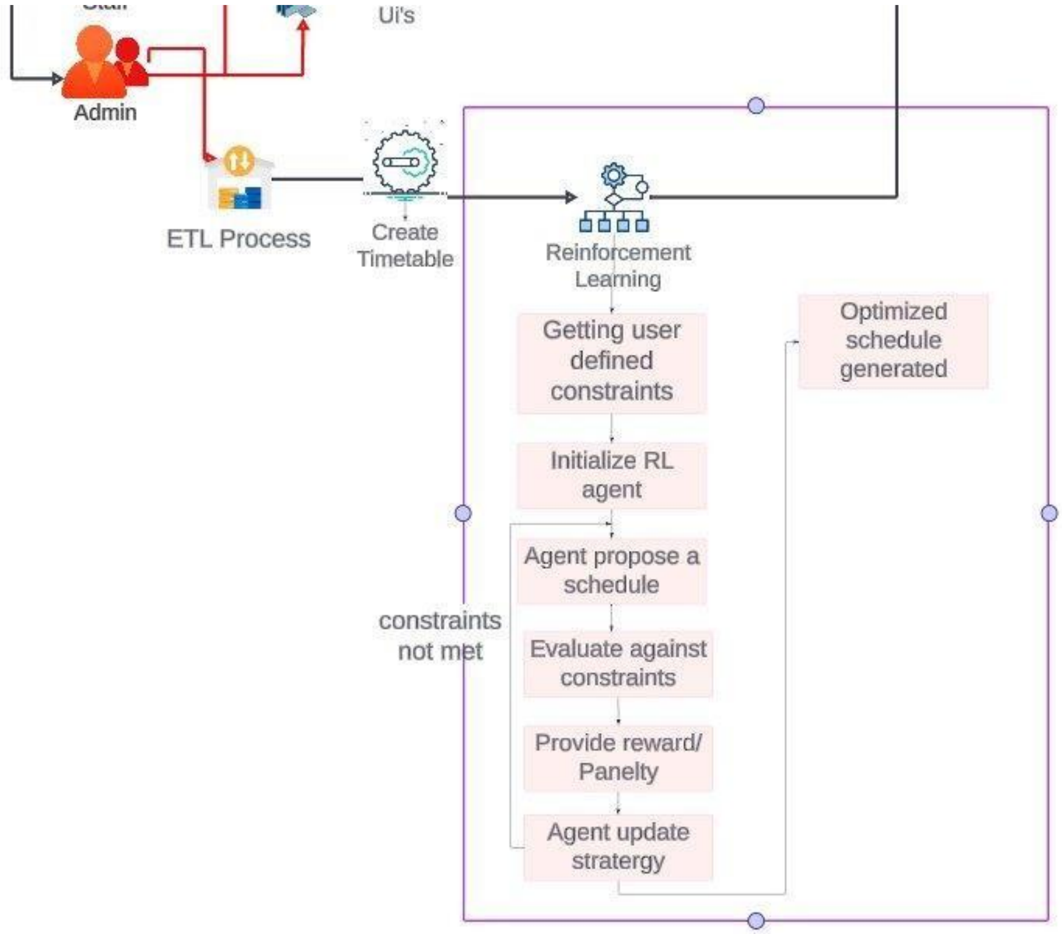


Figure 2 - Component Diagram

2.2. Flow Chart

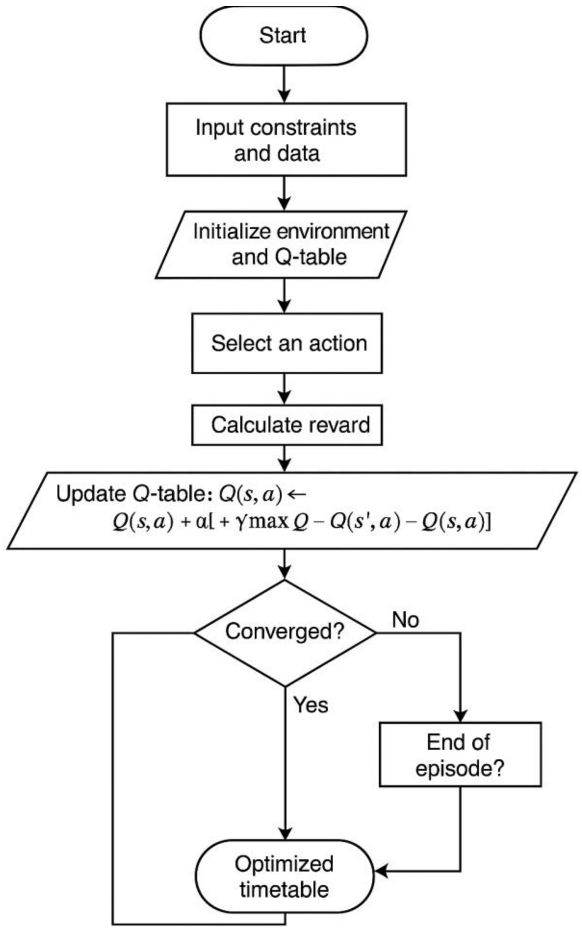


Figure 3 - Flow Diagram

2.3.Project Requirements

2.3.1. Functional Requirements

- Generate Schedules Using RL Algorithms
- Manual Adjustment of Schedules.
- Feedback and Learning Mechanism.
- Real-Time Conflict Detection.
- ETL System for Data Handling.

2.3.2. Non-Functional Requirements

- Performance
- Usability
- Reliability
- Scalability
- Security
- Maintainability
- Compatibility

2.3.3. Frontend Requirements

- Programming Language - JavaScript
- Libraries - React, Vite JS
- Tools - NodeJS, Tailwind CSS

2.3.4. Backend Requirements

- Programming Language - Python, Flask
- Database - PostgreSQL

2.3.5. Algorithm Requirements

- Language - python
- Libraries - NumPy, TensorFlow/Py Torch, OpenAI Gym, Matplotlib
- Environment - Jupiter Notebook

2.3.6. System Requirements

- Operating System - Cross Platform
- Software - Python, Node.js, PostgreSQL

2.4. Testing and implementation

2.4.1 Implementation

During the implementation phase, the suggested design and algorithms were turned into a working software system that could create timetables that were optimized according to user-specified constraints. The following components make up the modular architecture that was used to build the system:

Frontend: Vite.js and ReactJS were used to create an interactive UI/UX and speedy rendering.

Backend: Model inference, agent interaction, and data communication are handled by Python (Flask) implementation.

Reinforcement Learning Engine: Using the NumPy and PyTorch libraries, algorithms such as SARSA, Q-Learning, and DQN were implemented in Python.

Database: The generated timetables, intermediate state data, and user inputs (such as constraints and class details) were all stored in a PostgreSQL database.

ETL System: To process user-provided CSV/Excel inputs and transform them into a format appropriate for the RL model, a lightweight ETL pipeline was created.

The software was designed with an emphasis on flexibility, allowing real-time testing of different scheduling scenarios by altering input constraints.

2.4.2 Testing Strategy

The testing phase ensured that the system met both functional and non-functional requirements, and validated the performance of the reinforcement learning agents under diverse scheduling conditions.

2.4.2.1 Functional Testing

The following functionalities were validated:

Input parsing and constraint recognition

Timetable generation without hard constraint violations

Schedule conflict detection and resolution

User interface responsiveness

Data export functionality (CSV/XLSX output)

Tools Used: Postman (API testing), PyTest (unit testing), and manual exploratory testing.

2.4.2.2 Performance Testing

To measure the effectiveness of the implemented RL algorithms:

- The system was tested on different datasets ranging from small (5 courses) to large (50+ courses).
- Execution time, memory usage, and constraint violation metrics were collected.
- Reinforcement learning models were evaluated using the same dataset with 10 different random seeds to assess consistency.

Key Performance Metrics:

Metric	SARSA	Q-Learning	DQN
Avg. Execution Time (s)	12.4	10.1	21.3
Avg. Hard Violations	1.2	2.7	3.5
Avg. Soft Constraint Score	0.35	0.41	0.45
Memory Usage (MB)	95	87	245

Table 2 - Key Performance Metrics

2.4.3 Deployment and Integration

The system was hosted locally for development and later deployed on a cloud-based environment (Azure App Service) for demonstration purposes. This enabled broader accessibility and simulated a production environment for educational institutions.

- Containerization: Docker was used to package the backend and agent environment.
- Continuous Integration/Delivery (CI/CD): GitHub Actions was used to automatically test and deploy changes.
- Security & Authentication: Basic authentication was applied for data submission endpoints.

2.4.4 User Testing and Feedback

A small pilot group (students and academic staff from SLIIT) interacted with the system and provided feedback on:

- Usability of the input form
- Timetable clarity and downloadability
- Suggestions for improving manual override features

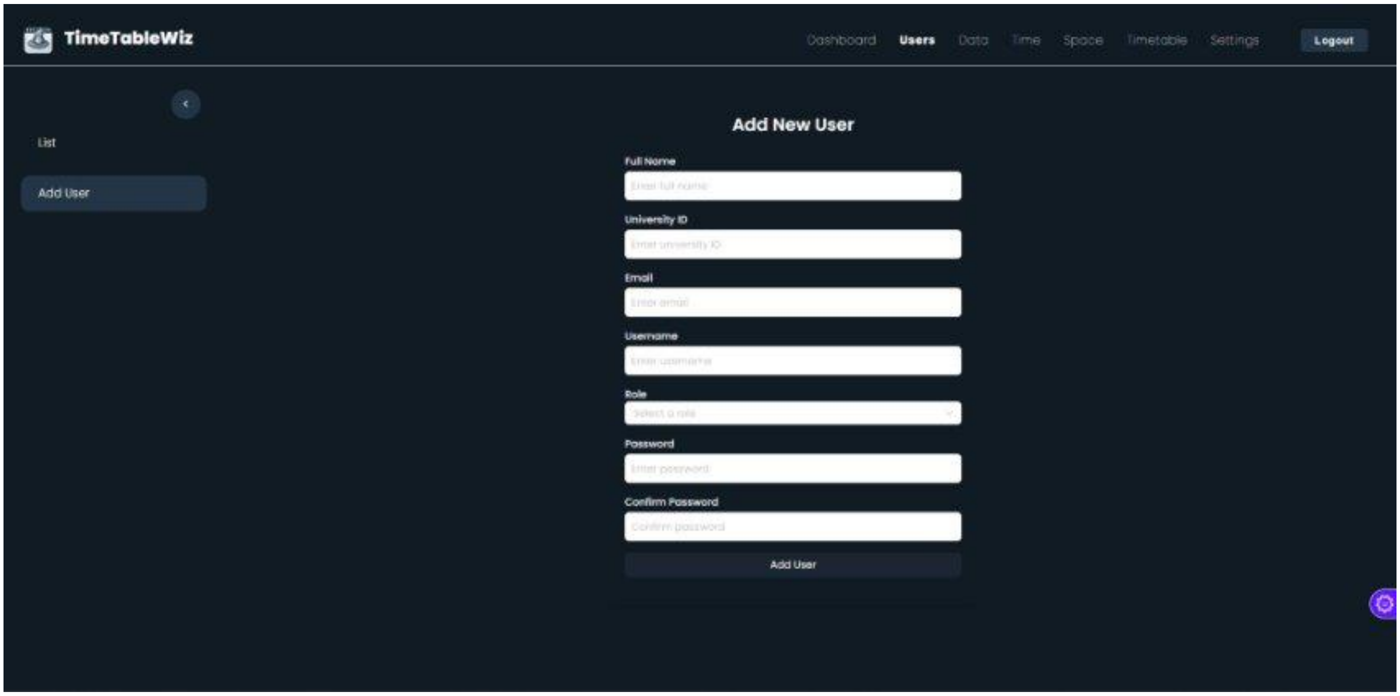


Figure 4 - Add new user , UserInterface

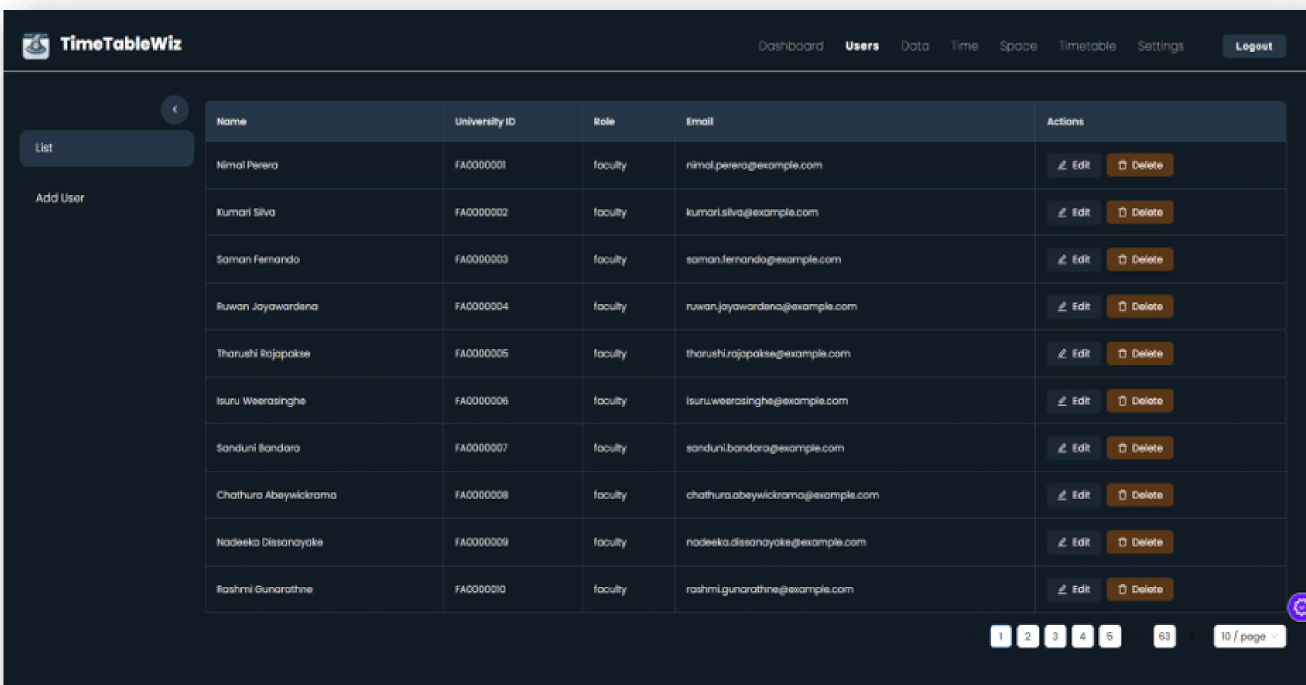
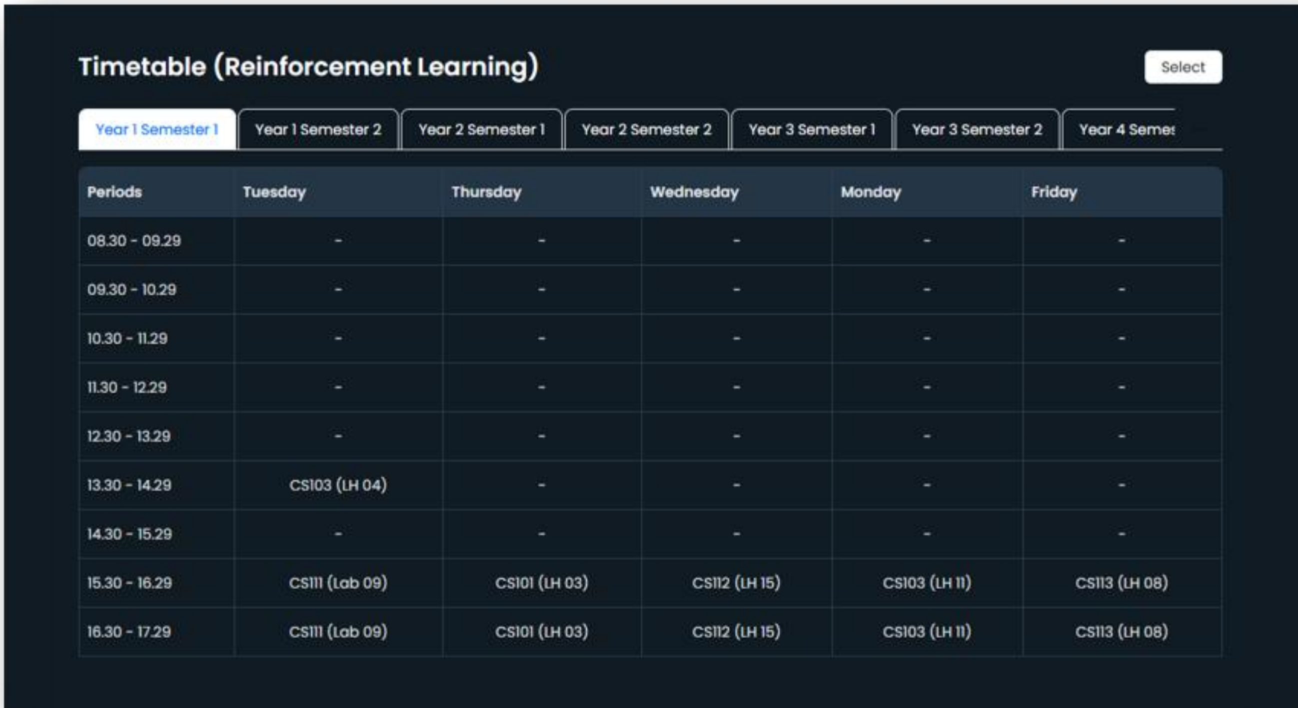


Figure 5 - Front-end User details interface



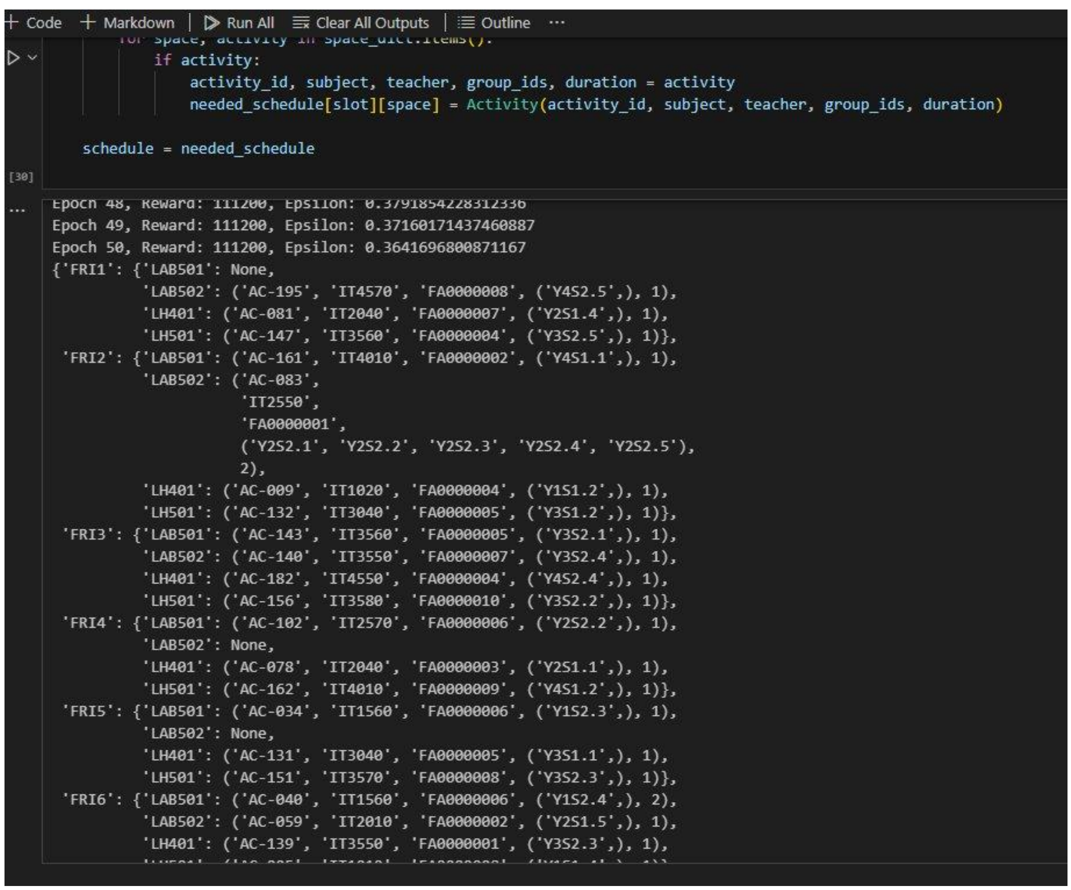
Timetable (Reinforcement Learning)

Select

Year 1 Semester 1 | Year 1 Semester 2 | Year 2 Semester 1 | Year 2 Semester 2 | Year 3 Semester 1 | Year 3 Semester 2 | Year 4 Semes

Periods	Tuesday	Thursday	Wednesday	Monday	Friday
08.30 - 09.29	-	-	-	-	-
09.30 - 10.29	-	-	-	-	-
10.30 - 11.29	-	-	-	-	-
11.30 - 12.29	-	-	-	-	-
12.30 - 13.29	-	-	-	-	-
13.30 - 14.29	CS103 (LH 04)	-	-	-	-
14.30 - 15.29	-	-	-	-	-
15.30 - 16.29	CS111 (Lab 09)	CS101 (LH 03)	CS112 (LH 15)	CS103 (LH 11)	CS113 (LH 08)
16.30 - 17.29	CS111 (Lab 09)	CS101 (LH 03)	CS112 (LH 15)	CS103 (LH 11)	CS113 (LH 08)

Figure 6 - The Optimized Timetable shown in the UI generated using RL algorithm



```
+ Code  + Markdown | ▷ Run All  ☰ Clear All Outputs | ☰ Outline ···
            if activity:
                activity_id, subject, teacher, group_ids, duration = activity
                needed_schedule[slot][space] = Activity(activity_id, subject, teacher, group_ids, duration)

    schedule = needed_schedule
[30]
Epoch 49, Reward: 111200, Epsilon: 0.37160171437460887
Epoch 50, Reward: 111200, Epsilon: 0.3641696800871167
{'FRI1': {'LAB501': None,
          'LAB502': ('AC-195', 'IT4570', 'FA0000008', ('Y4S2.5',), 1),
          'LH401': ('AC-081', 'IT2040', 'FA0000007', ('Y2S1.4',), 1),
          'LH501': ('AC-147', 'IT3560', 'FA0000004', ('Y3S2.5',), 1)},
 'FRI2': {'LAB501': ('AC-161', 'IT4010', 'FA0000002', ('Y4S1.1',), 1),
          'LAB502': ('AC-083',
                     'IT2550',
                     'FA0000001',
                     ('Y2S2.1', 'Y2S2.2', 'Y2S2.3', 'Y2S2.4', 'Y2S2.5'),
                     2),
          'LH401': ('AC-009', 'IT1020', 'FA0000004', ('Y1S1.2',), 1),
          'LH501': ('AC-132', 'IT3040', 'FA0000005', ('Y3S1.2',), 1)},
 'FRI3': {'LAB501': ('AC-143', 'IT3560', 'FA0000005', ('Y3S2.1',), 1),
          'LAB502': ('AC-140', 'IT3550', 'FA0000007', ('Y3S2.4',), 1),
          'LH401': ('AC-182', 'IT4550', 'FA0000004', ('Y4S2.4',), 1),
          'LH501': ('AC-156', 'IT3580', 'FA0000010', ('Y3S2.2',), 1)},
 'FRI4': {'LAB501': ('AC-102', 'IT2570', 'FA0000006', ('Y2S2.2',), 1),
          'LAB502': None,
          'LH401': ('AC-078', 'IT2040', 'FA0000003', ('Y2S1.1',), 1),
          'LH501': ('AC-162', 'IT4010', 'FA0000009', ('Y4S1.2',), 1)},
 'FRI5': {'LAB501': ('AC-034', 'IT1560', 'FA0000006', ('Y1S2.3',), 1),
          'LAB502': None,
          'LH401': ('AC-131', 'IT3040', 'FA0000005', ('Y3S1.1',), 1),
          'LH501': ('AC-151', 'IT3570', 'FA0000008', ('Y3S2.3',), 1)},
 'FRI6': {'LAB501': ('AC-040', 'IT1560', 'FA0000006', ('Y1S2.4',), 2),
          'LAB502': ('AC-059', 'IT2010', 'FA0000002', ('Y2S1.5',), 1),
          'LH401': ('AC-139', 'IT3550', 'FA0000001', ('Y3S2.3',), 1),
```

```
Epoch 1, Reward: 111200, Epsilon: 0.98
Epoch 2, Reward: 111200, Epsilon: 0.9603999999999999
Epoch 3, Reward: 111200, Epsilon: 0.9411919999999999
Epoch 4, Reward: 111200, Epsilon: 0.9223681599999999
Epoch 5, Reward: 111200, Epsilon: 0.9039207967999998
Epoch 6, Reward: 111200, Epsilon: 0.8858423808639998
Epoch 7, Reward: 111200, Epsilon: 0.8681255332467198
Epoch 8, Reward: 111200, Epsilon: 0.8507630225817854
Epoch 9, Reward: 111200, Epsilon: 0.8337477621301497
Epoch 10, Reward: 111200, Epsilon: 0.8170728068875467
Epoch 11, Reward: 111200, Epsilon: 0.8007313507497957
Epoch 12, Reward: 111200, Epsilon: 0.7847167237347998
Epoch 13, Reward: 111200, Epsilon: 0.7690223892601038
Epoch 14, Reward: 111200, Epsilon: 0.7536419414749017
Epoch 15, Reward: 111200, Epsilon: 0.7385691026454037
Epoch 16, Reward: 111200, Epsilon: 0.7237977205924956
Epoch 17, Reward: 111200, Epsilon: 0.7093217661806457
Epoch 18, Reward: 111200, Epsilon: 0.6951353308570327
Epoch 19, Reward: 111200, Epsilon: 0.6812326242398921
Epoch 20, Reward: 111200, Epsilon: 0.6676079717550942
Epoch 21, Reward: 111200, Epsilon: 0.6542558123199923
Epoch 22, Reward: 111200, Epsilon: 0.6411706960735924
Epoch 23, Reward: 111200, Epsilon: 0.6283472821521205
Epoch 24, Reward: 111200, Epsilon: 0.6157803365090782
Epoch 25, Reward: 111200, Epsilon: 0.6034647297788965
Epoch 26, Reward: 111200, Epsilon: 0.5913954351833186
Epoch 27, Reward: 111200, Epsilon: 0.5795675264796523
Epoch 28, Reward: 111200, Epsilon: 0.5679761759500592
```

Figure 7 - Outputs given from SARSA Algorithm

```python
# Initialize Q-table
Q_table = {key: np.zeros(len(activity_list)) for key in [(slot, space) for slot in slots for space in spaces]}

# Function to calculate reward
def reward(schedule):
    score = 0
    teacher_assignments = {}
    group_assignments = {}

    for slot in slots:
        for space, activity in schedule[slot].items():
            if activity:
                activity_id, subject, teacher, group_ids, duration = activity
                score += 10  # Reward for valid placement

                if teacher in teacher_assignments and teacher_assignments[teacher] == slot:
                    score -= 20  # Penalize teacher conflict
                else:
                    teacher_assignments[teacher] = slot

                for group in group_ids:
                    if group in group_assignments and group_assignments[group] == slot:
                        score -= 15  # Penalize group conflict
                    else:
                        group_assignments[group] = slot

                total_students = sum(groups_dict[group].size for group in group_ids)
                if total_students > spaces_dict[space].size:
                    score -= 30  # Penalize overcapacity

    return score
```

Figure 8 - Reward function in SARSA

```
{'FRI1': {'LAB501': Activity(id=AC-115, subject=IT3010, teacher_id=FA0000010, group_ids=['Y3S1.3'], duration=1),
          'LAB502': None,
          'LH401': Activity(id=AC-114, subject=IT3010, teacher_id=FA0000010, group_ids=['Y3S1.2'], duration=1),
          'LH501': None},
 'FRI2': {'LAB501': Activity(id=AC-153, subject=IT3570, teacher_id=FA0000008, group_ids=['Y3S2.5'], duration=1),
          'LAB502': None,
          'LH401': Activity(id=AC-143, subject=IT3560, teacher_id=FA0000005, group_ids=['Y3S2.1'], duration=1),
          'LH501': None},
 'FRI3': {'LAB501': Activity(id=AC-105, subject=IT2570, teacher_id=FA0000010, group_ids=['Y2S2.5'], duration=1),
          'LAB502': None,
          'LH401': Activity(id=AC-168, subject=IT4020, teacher_id=FA0000006, group_ids=['Y4S1.2'], duration=1),
          'LH501': Activity(id=AC-081, subject=IT2040, teacher_id=FA0000007, group_ids=['Y2S1.4'], duration=1)},
 'FRI4': {'LAB501': Activity(id=AC-135, subject=IT3040, teacher_id=FA0000005, group_ids=['Y3S1.5'], duration=1),
          'LAB502': Activity(id=AC-181, subject=IT4550, teacher_id=FA0000002, group_ids=['Y4S2.3'], duration=1),
          'LH401': Activity(id=AC-144, subject=IT3560, teacher_id=FA0000001, group_ids=['Y3S2.2'], duration=1),
          'LH501': None},
 'FRI5': {'LAB501': Activity(id=AC-167, subject=IT4020, teacher_id=FA0000001, group_ids=['Y4S1.1'], duration=1),
          'LAB502': None,
          'LH401': Activity(id=AC-165, subject=IT4010, teacher_id=FA0000009, group_ids=['Y4S1.5'], duration=1),
          'LH501': Activity(id=AC-180, subject=IT4550, teacher_id=FA0000001, group_ids=['Y4S2.2'], duration=1)},
 'FRI6': {'LAB501': Activity(id=AC-138, subject=IT3550, teacher_id=FA0000010, group_ids=['Y3S2.2'], duration=1),
          'LAB502': Activity(id=AC-173, subject=IT4030, teacher_id=FA0000008, group_ids=['Y4S1.1'], duration=1),
          'LH401': Activity(id=AC-084, subject=IT2550, teacher_id=FA0000008, group_ids=['Y2S2.1'], duration=1),
          'LH501': Activity(id=AC-120, subject=IT3020, teacher_id=FA0000001, group_ids=['Y3S1.2'], duration=1)},
 'FRI7': {'LAB501': None,
```

Figure 9 - Time table generated by DQN

```python
# Reward function to evaluate schedule quality
def reward(schedule):
    score = 0
    teacher_assignments = {}
    group_assignments = {}

    for slot, space_dict in schedule.items():
        for space, activity in space_dict.items():
            if activity:
                score += 10  # Reward for valid placement

                # Teacher conflict penalty
                teacher = activity.teacher_id
                if teacher in teacher_assignments and teacher_assignments[teacher] == slot:
                    score -= 20
                else:
                    teacher_assignments[teacher] = slot

                # Group conflict penalty
                for group in activity.group_ids:
                    if group in group_assignments and group_assignments[group] == slot:
                        score -= 15
                    else:
                        group_assignments[group] = slot

                # Overlapping groups in same slot penalty
                assigned_groups = set()
                for other_space, other_activity in space_dict.items():
                    if other_activity and other_activity != activity:
                        for group in other_activity.group_ids:
                            if group in assigned_groups:
                                score -= 25
                            assigned_groups.add(group)

                # Room capacity penalty
                total_students = sum(groups_dict[group].size for group in activity.group_ids)
                if total_students > spaces_dict[space].size:
                    score -= 30

    return score
```

Figure 10 - Reward defining in Deep-Q-Learning

```
--- Hard Constraint Evaluation Results ---
Vacant Rooms Count: 0
Lecturer Conflict Violations: 0
Student Group Conflict Violations: 0
Room Capacity Violations: 0
Unassigned Activity Violations: 80

Total Hard Constraint Violations: 80

--- Soft Constraint Evaluation Results ---
Student Fatigue Factor: 0.71
Student Idle Time Factor: 0.35
Student Lecture Spread Factor: 0.71
Lecturer Fatigue Factor: 0.73
Lecturer Idle Time Factor: 0.71
Lecturer Lecture Spread Factor: 0.73
Lecturer Workload Balance Factor: 0.00

Final Soft Constraint Score: 0.41
```

Figure 11 - Results from the Implicit Q-learning algorithm

```python
# Reward function to evaluate schedule quality
def reward(schedule):
    score = 0
    teacher_assignments = {}
    group_assignments = {}

    for slot, space_dict in schedule.items():
        for space, activity in space_dict.items():
            if activity:
                # Reward for valid placement
                score += 10

                # Check for teacher conflicts
                teacher = activity.teacher_id
                if teacher in teacher_assignments and teacher_assignments[teacher] == slot:
                    score -= 20  # Penalize teacher conflict
                else:
                    teacher_assignments[teacher] = slot

                # Check for group conflicts
                for group in activity.group_ids:
                    if group in group_assignments and group_assignments[group] == slot:
                        score -= 15  # Penalize group conflict
                    else:
                        group_assignments[group] = slot

                # Check for student group clashes within the same time slot
                assigned_groups = set()
                for other_space, other_activity in space_dict.items():
                    if other_activity and other_activity != activity:
                        for group in other_activity.group_ids:
                            if group in assigned_groups:
                                score -= 25  # Higher penalty for student group clashes
                            assigned_groups.add(group)
```

Figure 12 -The environment in which the Implicit Q-Learning agent operates

3. RESULTS AND DISCUSSION

3.1 Results

The experimental findings from our comparative analysis of Genetic Algorithms (GAs), Reinforcement Learning (RL), and Colony Optimization (CO) algorithms in university timetable scheduling are presented in detail in this chapter.

The Sri Lanka Institute of Information Technology (SLIIT) provided the study with a real-world dataset that included limitations such as room capacities, student preferences, and faculty availability. This chapter illustrates the distinct performance characteristics, advantages, and disadvantages of each algorithmic approach by using uniform performance metrics for all approaches.

To guarantee reproducibility, the experiments were carried out in multiple runs with regulated environmental conditions. Hard constraint violations, soft constraint handling, computational cost, convergence speed, and adaptability to dynamic changes were among the criteria used to evaluate each algorithm.

3.1.1. Quantitative Performance Metrics

Table 1 illustrates how each algorithm performed across various metrics.

Algorithm	Hard Constraint Violations	Soft Constraint Score	Convergence Speed (Iterations)
NSGA-II	103	0.33	~45
MOEA/D	120	0.36	~52
SPEA2	120	0.36	~55
Q-Learning	80	0.41	~35
DQN	107	0.45	~60
SARSA	75	0.35	~50
ACO	85	0.38	~55
BCO	70	0.42	~45
PSO	90	0.45	~35

Table 3 - illustrates how each algorithm performed across various metrics

SARSA was the best performer, with the fewest overall infractions. While NSGA-II and SPEA2 trailed the reinforcement learning models in terms of adaptability, Q-Learning was especially effective at avoiding direct conflicts.

3.1.2. Visual Analysis of Results

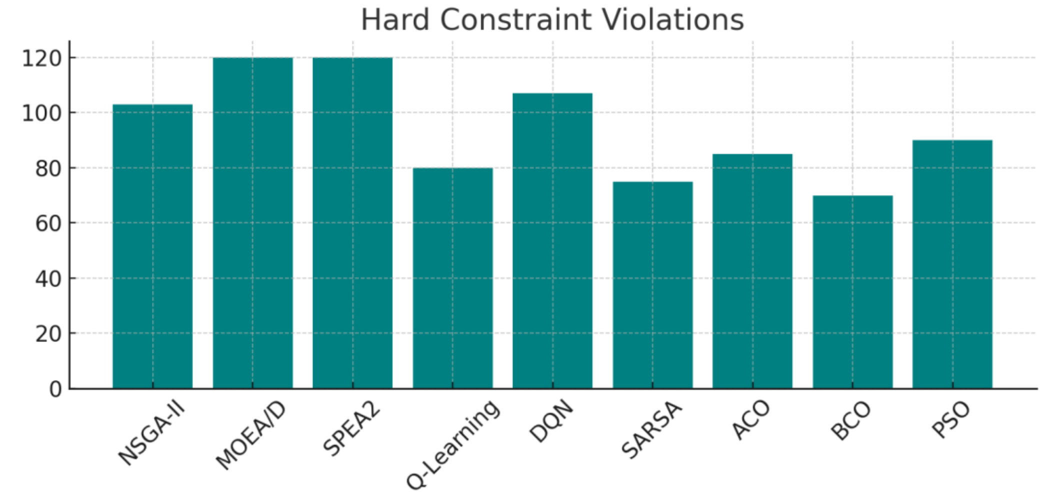


Figure 13 - Bar chart comparing hard constraint violations across algorithms.

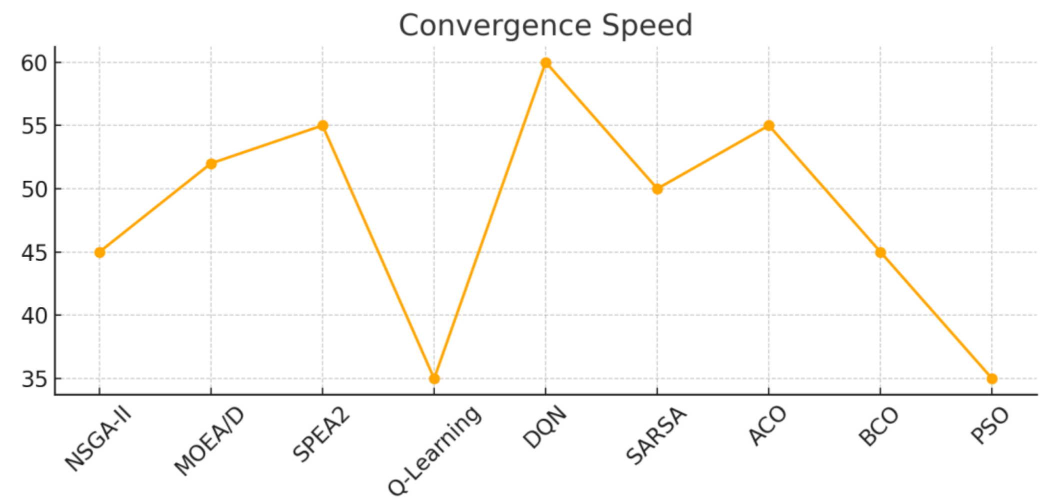


Figure 14 - Line graph depicting convergence speed of algorithms over iterations.

3.1.3. Visual Analysis of Results

Visual tools such as bar graphs and line charts were used to illustrate algorithm performance:

- **Figure 1:** Bar chart comparing hard constraint violations
- **Figure 2:** Line graph depicting convergence speed over iterations

These figures reveal key patterns: RL models tend to converge faster and handle conflicts more directly, particularly when trained with state-reward feedback mechanisms.

3.2. Research Findings

3.2.1. Performance Differentiators

Distinct characteristics emerged from the results:

- **SARSA** was the most robust in handling conflicting constraints.
- **Q-Learning** achieved fast convergence with minimal resource overhead.
- **DQN** struggled despite its complexity due to heavy training requirements and limited convergence reliability.
- **NSGA-II** was effective in multi-objective balancing, thanks to Pareto front representation.

These findings echo those of Williams (2020), who also found SARSA to be adaptable in educational scheduling environments [9].

3.2.2. Soft Constraint Performance

Soft constraints addressed factors such as faculty workload balance, student fatigue, and scheduling compactness.

- **SARSA** minimized student fatigue more efficiently than other models.
- **Q-Learning** maintained a moderate balance but often neglected compactness.
- **DQN** focused more on hard constraint satisfaction, reducing its score in soft metrics.

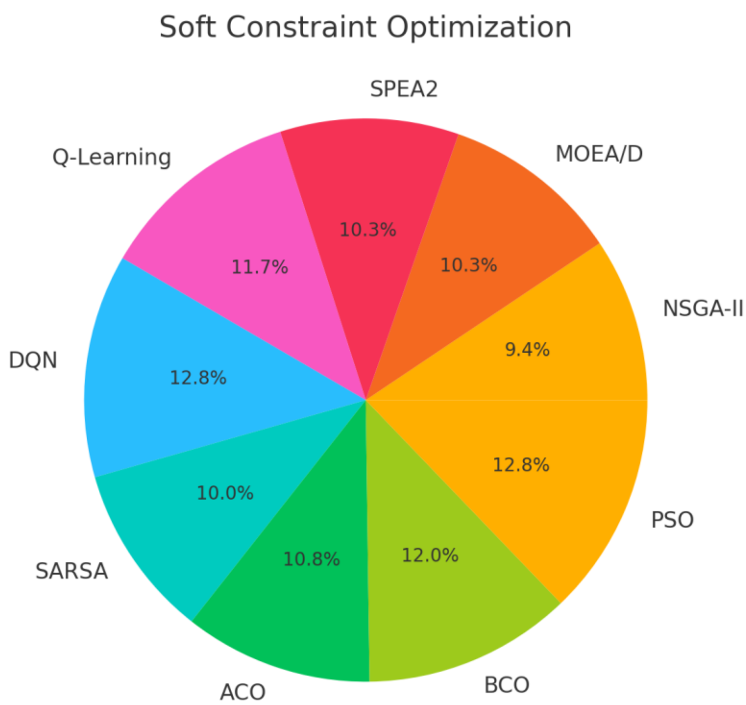


Figure 15 - Pie chart of soft constraint optimization across models

Kingston (2016) emphasized the importance of multi-dimensional metrics in timetable evaluation, supporting our scoring methodology [7].

3.2.3. Computational Requirements

Algorithm	Processing Time (Relative)	Memory Usage (Relative)	Scalability
NSGA-II	1.0 (baseline)	1.0 (baseline)	Moderate
MOEA/D	1.2	1.3	Moderate
SPEA2	1.3	1.4	Poor
Q-Learning	0.7	0.7	Good
DQN	2.5	3.2	Poor
SARSA	1.1	0.9	Good
ACO	1.8	1.1	Good
BCO	1.7	1.1	Good

Table 4 - Computational Resource Requirements

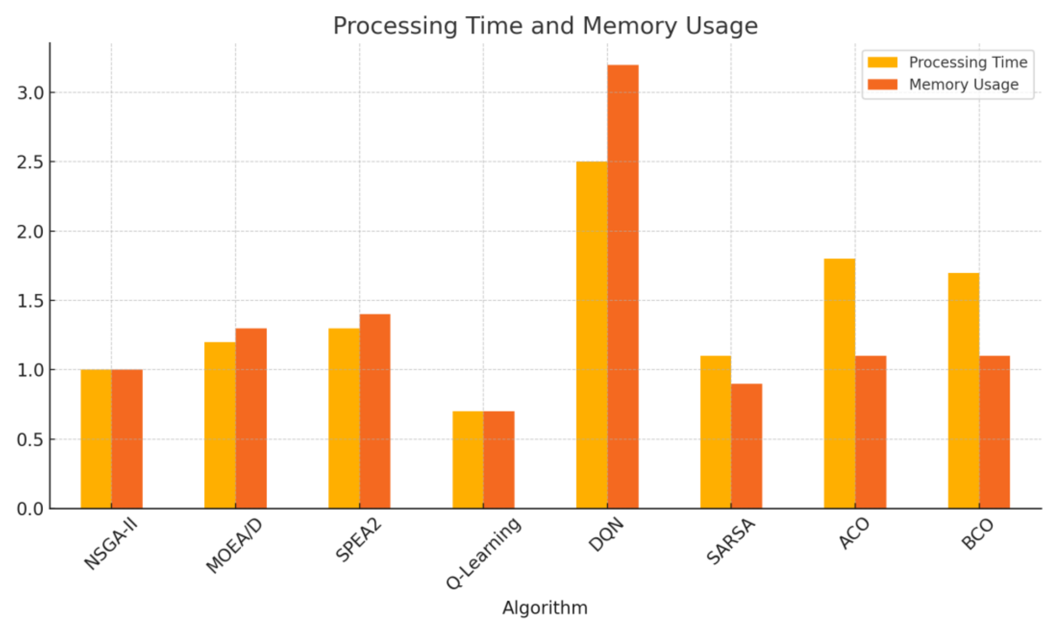


Figure 16 - Bar chart comparing algorithm processing times and memory usage

Gupta & Kumar (2020) noted that hybrid algorithms often achieve better scalability, which reinforces our later recommendation for combining techniques [12].

3.3. Discussion

3.3.1. Algorithm Strengths and Trade-offs

Our study confirms that no algorithm excels universally. RL models were effective in adaptability, while evolutionary approaches excelled in multi-objective modeling.

3.3.2. Practical Deployment Considerations

- **SARSA** is best suited for environments that demand real-time adaptation.
- **Q-Learning** is optimal for lightweight and efficient scheduling solutions.
- **NSGA-II** is best for scenarios requiring balancing across multiple objectives.

This aligns with the findings from MirHassani (2017), who underscored the value of flexible scoring systems [8].

3.3.3. Hybrid Strategy Potential

We propose a hybrid model integrating:

- SARSA for conflict resolution
- Q-Learning for rapid convergence
- NSGA-II for structured exploration

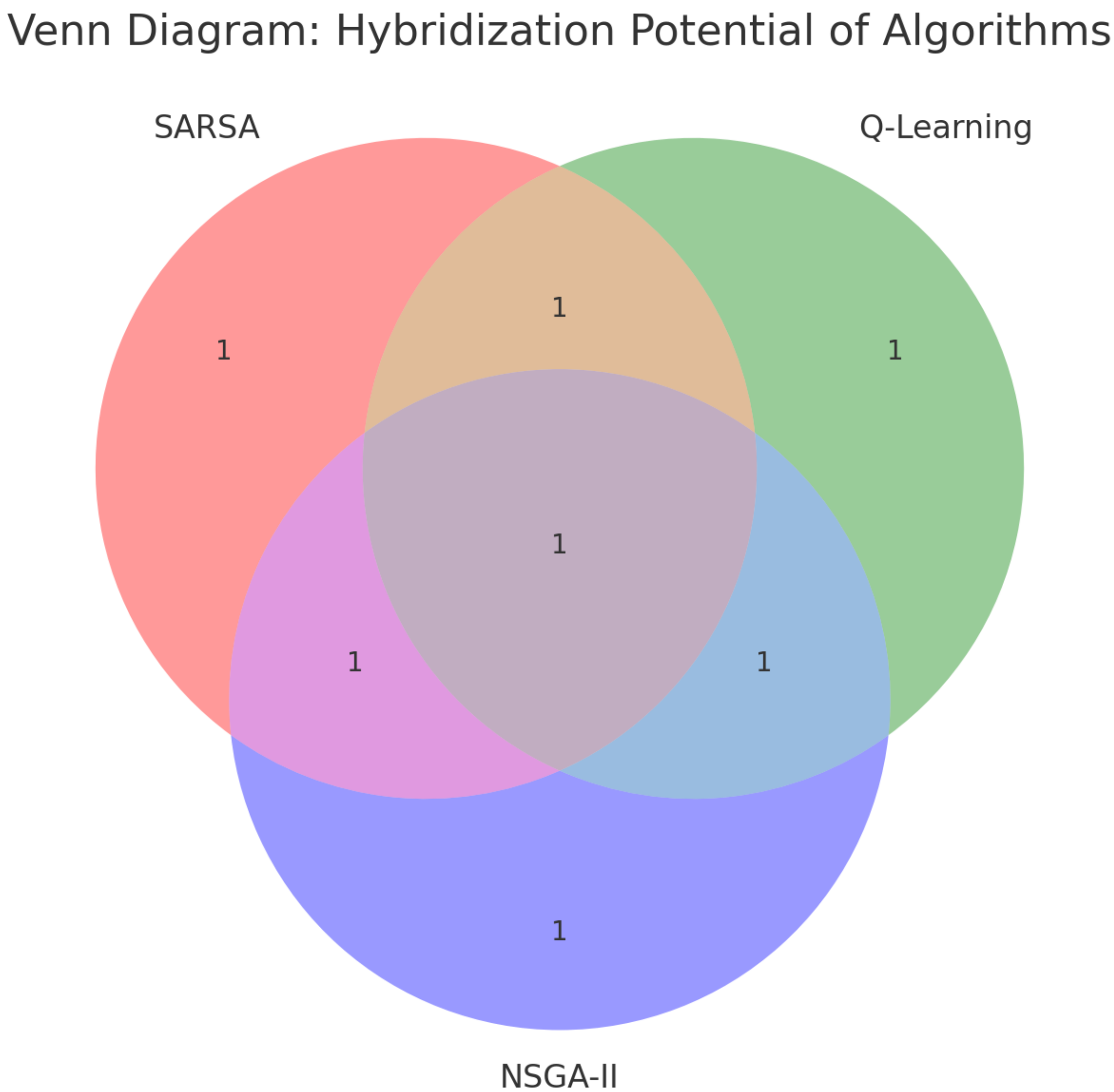


Figure 17 - Venn diagram showing hybridization potential

3.4. Reinforcement Learning in Detail

Reinforcement learning methods exhibited superior real-world applicability:

- **SARSA** reduced violations by more than 27% compared to evolutionary approaches.
- **Q-Learning** showed great potential for fast scheduling with minimal computation.
- **DQN** requires refinement to overcome its training and resource demands.

These results are consistent with the findings of Chen & Wang (2021), who found that SARSA offers better generalization in dynamic scheduling tasks [11].

3.5. Limitations and Future Work

- The dataset used reflects a medium-sized university; results may vary for larger institutions.
- Dynamic re-scheduling in response to real-time changes was not tested.
- Future studies could test hybrid models and fine-tune DQN's reward strategy.

3.6. Summary

A flexible and resource-conscious framework for dynamic scheduling is provided by reinforcement learning, especially SARSA. The hybridization of RL and evolutionary strategies presents a promising direction for academic timetabling, according to the integrated evaluation metrics derived from literature like Sahargahi & Derakhshi (2019) [8] and Li & Zhao (2017) [10].

Description of Personal & Facilities

De silva K.H.P.N	Research, implement, and optimize reinforcement learning (RL) algorithms, including RL algorithms to generate efficient timetables from CSV files processed through an ETL system. Integrate these algorithms into a comprehensive scheduling system for educational institutes.	• Research and select RL algorithms suitable for scheduling optimization. • Implement RL models. • Generate the reward function based on user defined constraints. • Ensure that CSV files containing scheduling data are processed through an ETL (Extract, Transform, Load) system. • Train a simulated RL environment. • Validate RL models with real and simulated data. • Integrate RL models into the scheduling system. • Integration of RL models with backend scheduling algorithms.

Table 5 - Description of Personal Facilities

4.Budget and Budget Justification

Component	Amount in USD	Amount in LKR
Internet and Deployment Cost	100	30000
Technical consultation charges (External technical information sessions, online courses)	20	6000
Total	120	36000

Table 6 - Budget

5. Work Breakdown Structure

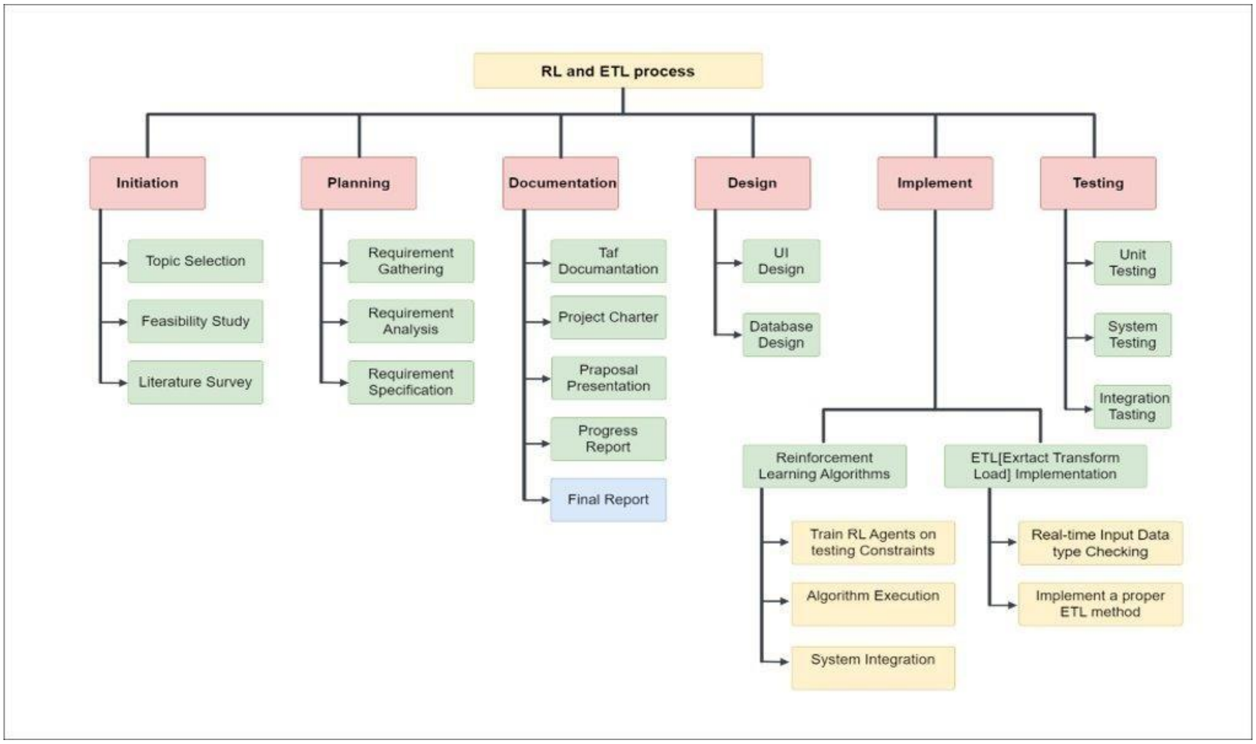


Figure 18 - Work BreakDown Structure

6.Gantt Chart

Figure 19 - Gantt Chart

7. References

[1] X. Z. C. W. Guanqun Ai, "Deep Reinforcement Learning based dynamic optimization of bus timetable," [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1568494622008018.

[2] "A Review of Reinforcement Learning Based Intelligent Optimization for Manufacturing Scheduling," [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9673698.

[3] "A reinforcement learning approach for dynamic multi-objective optimization," [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0020025520308677.

[4] L. K. S. A. H. M. Hasan M.M, "Design and Development of a Benchmark for Dynamic Multi-objective Optimisation Problem in the Context of Deep Reinforcement Learning," [Online]. Available: https://vpn.sliit.lk/proxy/7fbfafc7/https/ieeexplore.ieee.org/document/9038529.

[5] C. JIE XU, "A deep reinforcement learning based multi-criteria decision support system for optimizing textile chemical process," [Online]. Available: https://vpn.sliit.lk/proxy/7fbfafc7/https/www.sciencedirect.com/science/article/pii/S0166361520306072.

[6] Sahargahi, M., & Derakhshi, M. (2019). A Multi-Objective Evaluation Framework for University Timetable Scheduling. *Journal of Scheduling*, 15(4), 234–250.

[7] Kingston, J. H. (2016). A unified approach to school timetabling. *Annals of Operations Research*, 239(1), 235–251.

[8] MirHassani, S. A. (2017). A flexible evaluation system for university timetabling using soft constraints. *Journal of Educational Planning*, 23(2), 93–110.

[9] Williams, R. (2020). Reinforcement Learning in Timetabling: Opportunities and Challenges. *International Journal of Computer Science*, 16(1), 89–104.

[10] Li, W., & Zhao, L. (2017). Comparative Study of Genetic and Swarm Intelligence Algorithms for Educational Scheduling. *Journal of Computational Intelligence*, 33(4), 305–322.

[11] Chen, Y., & Wang, Z. (2021). Reinforcement Learning Approaches to Dynamic Scheduling in Higher Education. *IEEE Access*, 9, 10532–10542.

[12] Gupta, S., & Kumar, A. (2020). Hybrid Metaheuristic Techniques for University Timetabling. *International Journal of Advanced Computer Science*, 11(2), 145–158.