

Timetable Wiz: Advanced Timetable Generation Solution for Educational Institutions

Kariyawasm Don Easara Ivanjaya Weerasinghe

(IT21259852)

BSc (Hons) Degree in Information Technology Specialization in Information
Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

Timetable Wiz: Advanced Timetable Generation Solution for Educational Institutions

Kariyawasm Don Easara Ivanjaya Weerasinghe

(IT21259852)

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of Science
Special Honors Degree in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

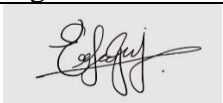
Sri Lanka

April 2025

DECLARATION

I declare that this is my own work and this dissertation 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.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)

Name	Student ID	Signature
Weerasinghe K.D.E. I	IT21259852	

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

Signature of the Supervisor

(Mr. Jeewaka Perera)

Date

Signature of the Co-Supervisor

(Ms.Sasini Hathursinghe)

Date

ABSTRACT

Time is a crucial resource in both personal and organizational contexts. Effective time management directly influences productivity, efficiency and overall success of any task, in educational institutions- efficient time management is vital to ensure smooth operations. scheduling plays a vital role in this process, in educational setting these could be scheduling lectures, examinations, assignments and events, However, scheduling is known to be a complex task, often classified as Non-deterministic Polynomial-time hard problem (NP-hard), which require consideration of numerous factors such as, Lecturer Availability, student preferences, hall availability to name a few, Existing systems have been identified to rely on manual methods and have taken heuristic based approaches, which have been identified to fall short of addressing complexities in most optimal way. To address this NP-hard Problem, this research aims to develop an advanced scheduling solution that leverages multiple optimization techniques, including Evolutionary Algorithms (EAs), Reinforcement Learning (RL) colony optimization and evaluation criteria. A key component in this study is the Implementation of Evolutionary Algorithms, which are tailored to educational timetabling, This involves the research and design of Evolutionary algorithms which gathers constraints through a user-friendly interface, allowing users to input both soft and hard constraints relevant to development of timetable, furthermore the research is aiming to explore and compare variants of Evolutionary algorithms (EAs) ,Such as Genetic Algorithms (GAs) , Non-Dominated-Sorting-Genetic algorithm (NSGA-II),Strength Pareto Evolutionary Algorithm 2 (SPEA2) , Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) to determine most effective approaches for timetable generation. Additionally, the Software development will integrate a Role-based-Access Control (RBAC) system to ensure secure access and to provide own user perspectives, furthermore the system will also offer customized timetable views and Application programming interface (Api) for seamless integration with personal or institutional calendar applications

Keywords – non-deterministic-polynomial time hard problem (NP-Hard), Evolutionary Algorithms (EAs), Genetic Algorithms (GAs), Non-dominated sorting Genetic Algorithm II(NSGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEAD2), Multi-objective Algorithm based on Decomposition (MOEA/D), Reinforcement Learning (RL) , Colony Optimization (CO), Role-based Access Control (RBAC)

ACKNOWLEDGEMENT

First, I would like to express my deep gratitude Mr. Jeewaka Perera Senior Lecturer Faculty of Computing, Sri Lanka Institute of Information Technology, our research supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this proposal work.

I would also like to thank Ms. Sasini Hathurusinghe, our co-supervisor, for her advice and assistance in delivering the proposal report. My grateful thanks are also extended to CDAP team for their guidance, concerns and help in delivering a proposal which covers all the required aspects.

I would also like to extend my thanks and gratitude to the colleagues of the research group for their support and great teamwork. Finally, I wish to thank my parents for their support and encouragement throughout this process.

TABLE OF CONTENT

DECLARATION	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENT	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	ix
1 INTRODUCTION	1
1.1 Background Literature	1
1.1.1 Literature Survey	3
1.1.2 Background Survey	10
1.1.3 Research Gap	12
1.1.4 Research Problem	15
1.2 Research Objectives	16
1.2.1 Specific Objectives	16
2 SYSTEM METHODOLOGY	18
2.1 Requirements	18
2.1.1 Functional Requirements	19
2.1.2 Non-Functional Requirements	20
2.1.3 Use Case Diagram	21
2.2 System Overview	22
2.3 Solution Representation	23
2.4 Algorithms	26
2.4.1 Problem Definition	26
2.4.2 Common Implementation Aspects:	27
2.4.3 NSGA-II:	27

2.4.4	MOEA/D:	30
2.4.5	SPEA2:	31
2.5	Experimental Setup:	34
2.5.1	Dataset Descriptions:	34
2.5.2	Algorithm Settings:	36
2.5.3	System UI/UX and Usability:	39
2.5.4	System Requirements	44
2.5.5	Challenges	44
2.6	Work Breakdown Structure	45
2.7	Gantt Chart	45
2.8	Commercialization aspects of the project	46
2.9	Testing and Implementation	49
2.9.1	Functional Testing	52
2.9.2	Non-Functional Testing	53
3	Results & Discussion	54
3.1.1	Results	54
3.1.2	Research Findings	59
3.1.3	Discussion	63
4	Conclusion	67
	REFERENCES	71
5	APPENDICES	74

LIST OF FIGURES

Figure 1.1: Classification of Evolutionary Algorithms Selected [7]	1
Figure 1.2: Role Based Access Control Timetable System	2
Figure 1.3: Flowchart Depicting Genetic Algorithms workflow [1]	3
Figure 1.4: Roulette Wheel Selection [1]	4
Figure 1.5: Tournament Selection Mechanism [1]	4
Figure 1.6: Crossover Method [1]	5
Figure 1.7: Performance Results [2]	6
Figure 1.8: Chromosome Representation in GA [2]	7
Figure 1.8: NSGA-II VPS Procedure [4]	9
Figure 1.9: Comparison between NSGA-II vs NSGA-II VPS [4]	10
Figure 1.10: Student Background Survey Results	11
Figure 2.1: Use Case Diagram	21
Figure 2.4: High-Level Diagram for overall system diagram	22
Figure 2.4: Chromosome Representation for Each Dataset	24
Figure 2.5: Chromosome Representation Choice	25
Figure 3.3: Timetable Generation via Selected Algorithms	40
Figure 3.4: Timetable Results via Selected Algorithms and Exported HTML format	41
Figure 3.3: Role Base Access Views for Each user and Journey	43
Figure 3.1 : Hard Constraint Violations	54
Figure 3.2: Soft Constraint Scores	55
Figure 3.3: Hypervolume Progression	57
Figure 3.4: Final Performance Metrics	58

LIST OF TABLES

Table 1.1: Comparison of Existing System	14
Algorithm 01 NSGA-II:	28
Algorithm 02 MOEA/D	30
Algorithm 03 SPEA2:	32
Table 3.4: Subscription plan of 'TimeTableWiz Application' application	47
Table 3.5: Budget Plan per Month	48
Table 3.6: TimetableWiz test case 1: NSGA-II for SLIIT dataset	49
Table 3.7: TimetableWiz test case 2: for RBAC	50
Table 3.8: TimetableWiz test case 3: for System Integration Timetable Views.	50
Table 3.9: TimetableWiz test case 4: for faculty unavailability test case	51
Table 3.1: Algorithm comparison in SLIIT Dataset.	59
Table 3.2: Algorithm comparison in Muni Dataset.	60

LIST OF APPENDICES

Appendix A Logo _____	74
Appendix B Gantt Chart _____	74
Appendix C Work Breakdown Chart _____	75
Appendix D Questionnaire for Students Regarding Timetable _____	76
Appendix E Questionnaire for Academic Staff Regarding Timetable _____	81
Appendix F Questionnaire for Administration Regarding Timetable _____	87
Appendix G System Workflow Diagram _____	91
Appendix H Turnitin Report _____	92

LIST OF ABBREVIATIONS

Abbreviation	Description
GA	Genetic Algorithms
EA	Evolutionary Algorithms
NSGA-II	Non-dominated Sorting Genetic Algorithm II
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
SPEA2	Strength Pareto Evolutionary Algorithm 2
RBAC	Role-Based Access Control
SDLC	Software Development Life Cycle
API	Application Programming Interface
SLIIT	Sri Lanka Institute of Information Technology
ITC	International Timetabling Competition
MUNI	muni-fsps-spr17 Dataset via ITC

1 INTRODUCTION

1.1 Background Literature

In the domain of educational scheduling, the complexity of generating optimal timetables is well-known [1], when considering the numerous constraints and dynamic factors involved [1]. Traditional methods, such as manual scheduling or heuristic approaches, often fall short in addressing the multifaceted challenges presented by this task [1][3][9]. As educational institutions continue to grow in size and complexity, the demand for well-refined timetable systems grows.

Evolutionary Algorithms (EAs) have emerged as a promising solution in this context, offering a robust method for exploring vast solution spaces and optimizing scheduling problems [8]. EAs are inspired by the principles of natural selection and evolution, utilizing processes such as selection, crossover, and mutation to evolve solutions over successive generations. This adaptability makes them particularly well-suited for solving NP-hard problems like timetable scheduling [1][2].

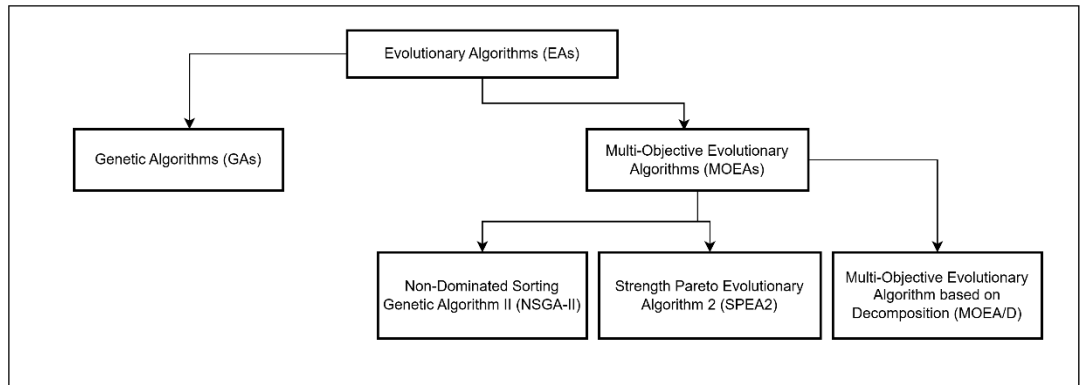


Figure 1.1: Classification of Evolutionary Algorithms Selected [7]

This research specifically focuses on the implementation of EAs tailored to the educational timetable problem. The study will explore and compare various EA variants, including Non-Dominated Sorting Genetic Algorithm II (NSGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEA2), and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D). Each of these variants offers distinct advantages in handling the multi-objective nature of scheduling, where multiple conflicting criteria must be balanced. In addition to the exploration of EAs, the

research also delves into the software development aspects necessary for the successful deployment of a scheduling system. This includes the integration of a Role-Based Access Control (RBAC) system to ensure secure and personalized access to the scheduling platform, as well as the development of APIs for seamless integration with existing calendar applications. The system is designed to be user-friendly, with an interface that allows users to input hard and soft constraints, thus enabling the Algorithms to generate schedules that meet the specific needs of the institution.

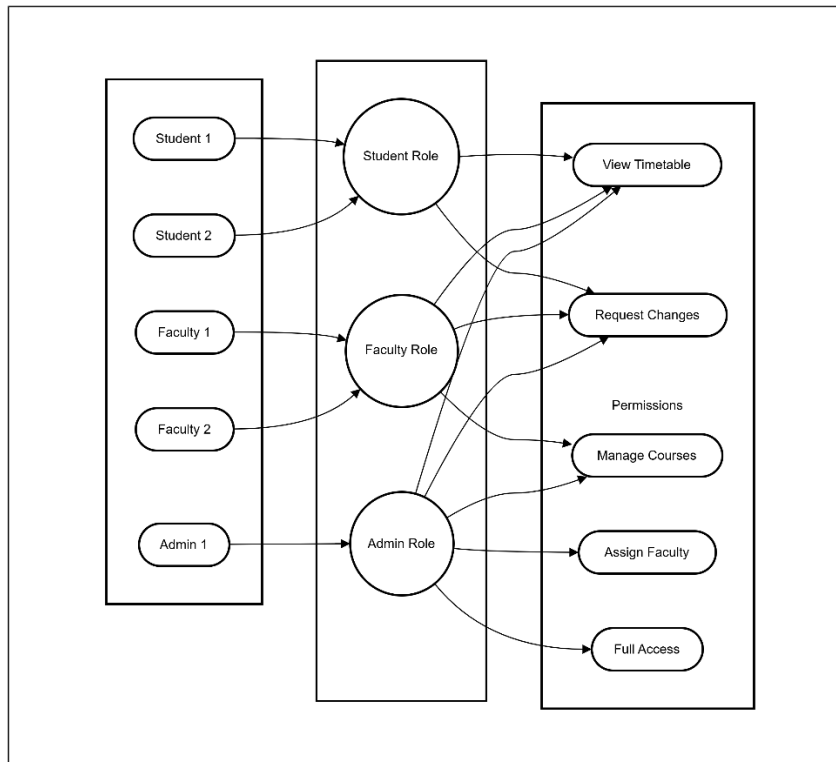


Figure 1.2: Role Based Access Control Timetable System

The study aims not only to advance the state of the art in educational scheduling but also to provide a comprehensive framework for future research in the field. By combining advanced optimization techniques with solid software engineering principles, this research seeks to develop a scalable, flexible, and efficient scheduling solution that can be adapted to the unique requirements of various educational institutions

1.1.1 Literature Survey

Genetic Algorithms for University Course timetabling problem [1]

In this Research it was aimed to demonstrate the usefulness of genetic algorithms usage to obtain optimal solutions in general timetable scheduling; while recognizing commercial software availability, this research addresses its lack of generality rarely meeting the demand of various institutions, therefore the requirement of specific coding as per respective universities

Furthermore, Research Discussed about other methods of automated timetabling such as Graph coloring algorithms, mathematical programming algorithms, use of database management systems, and further discussed about how genetic algorithms are good at figuring out through vast solutions spaces

The research highlights that genetic algorithms are particularly adept at navigating vast solution spaces, making them well-suited for complex timetable problems where other methods may fall short.

Genetic algorithms are optimization techniques inspired by natural selection, aiming to find approximate solutions to complex problems. The process involves several key steps

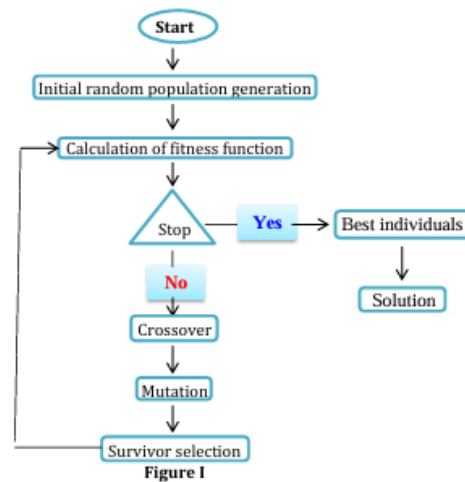
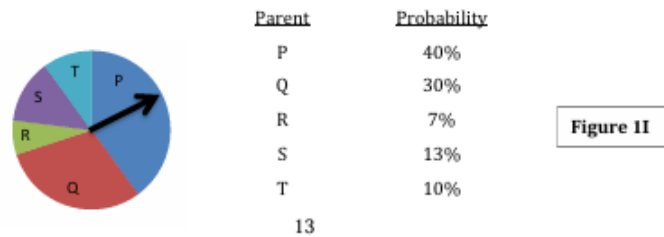


Figure 1.3: Flowchart Depicting Genetic Algorithms workflow [1]

The research extensively discusses various selection methods used in genetic algorithms, including:

- **Roulette Wheel Selection:** Also known as fitness proportionate selection, this method assigns selection probabilities based on fitness scores. It ensures that higher-fit individuals have a greater chance of being selected, though it can lead to premature convergence if not managed properly.

2.5.2 Selection based on the wheel of fortune.



- **Tournament Selection:** This method involves randomly selecting a subset of the population to compete for inclusion in the next generation. The individual with the highest fitness in the subset is selected. Tournament selection is known for its simplicity and efficiency, making it suitable for parallel and non-parallel implementations.
- **Stochastic Tournament Selection:** An extension of tournament selection that introduces randomness into the selection process, allowing for a more nuanced balance between exploration and exploitation.



Figure 1.5: Tournament Selection Mechanism [1]

Additionally, Research Discusses about Cross-Over Methods such as one-point crossovers and uniform crossovers, and discusses the importance of how selecting random point increases variety for the next generation

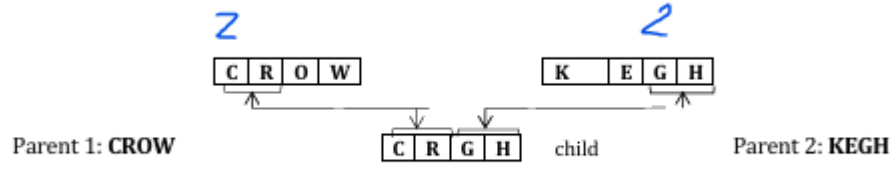


Figure 1.6: Crossover Method [1]

In the Initialization Phase Research Discusses about Clashes method for violating hard constraints in timetable generation, each instance of constraint violation was recorded as a clash, and the total number of clashes was used to calculate the fitness value of the timetable solution

$$f = \frac{1}{c + 1}$$

Where:

- $f = \text{Fitness Value}$
- $c = \text{NumberOfConstraintViolations(clashes)}$

In this research, Research Was Specifically focused on using Tournament selection and one point crossover method key findings are as follows

- As the population size increases, the number of generations required to reach an optimal solution also increases. This implies that larger populations may result in more iterations to reach a valid solution.
- An increase in the mutation rate leads to a higher number of clashes. The optimal mutation rate for minimizing clashes was found to be 0.01, as higher mutation rates increase the number of constraint violations.
- As the mutation rate increases, the fitness value decreases. This indicates that higher mutation rates reduce the overall quality of the solutions, with the optimal mutation rate again being 0.01.

Timetabling with Three-Parent Genetic Algorithms: A Preliminary Study [2]

In this paper emphasizes while two parent Genetic Algorithms are common, paper explored a multi parent approach to the timetabling problem, offering effective solutions to manage complex scheduling constraints

The uniform three parent crossover technique is introduced selecting three parent solutions and creating offspring based on bit comparisons, specifically if corresponding bits from the first two parents match, they are used in the offspring, otherwise, the bit from the third parent is selected, this method was aimed at incorporating diverse genetic material, potentially leading to more robust solutions

Key findings and results as follow,

With a population size of 100 and a mutation rate of 0.01, the two-parent scheduler achieved a timetable with zero clashes and a fitness value of 1. In contrast, the three-parent schedule produced a timetable with ten clashes and a fitness value ranging from 1 to 0.25. The three-parent approach reached an optimal solution on the 14th run, while the two-parent approach reached it on the 15th run.

Table 1: (Test 1): The number of generations the three-parent course scheduler takes to reach an optimal solution was tested.

Number of runs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Generations	816	427	31	1000	193	411	1000	1000	1000	860	972	1000	1000	34	1000	453	283	1000	893	77
Fitness value	1	1	1	0.5	0	0	1	1	3	0	0	1	1	0	1	0	0	1	0	0
Number of clashes	0	0	0	1	0	0	1	1	3	0	0	1	1	0	1	0	0	1	0	0

Figure 1.7: Performance Results [2]

Increasing the population size to 2000, while keeping the mutation rate and other parameters constant, showed that the three-parent scheduler required fewer generations to reach a solution compared to the two-parent scheduler.

When the mutation rate varied from 0.01 to 0.20, the three-parent scheduler resulted in only three clashes, whereas the two-parent scheduler exhibited 74 clashes. This indicates that the three-parent approach is more effective in managing constraint violations under different mutation rates.

An analysis of mutation rate versus fitness value revealed that the three-parent scheduler maintained a constant fitness value during the initial 10 runs, but the fitness

rapidly decreased after the 21st run. Conversely, the two-parent scheduler showed a more consistent decline in fitness.

Genetic Algorithms for Solving University Course Timetabling problem using dynamic chromosomes [3]

This Research, while Agreeing to the fact that scheduling is a np-hard problem, introduces a dynamic chromosome size, which adapts to the varying number of courses within each departments making the algorithm applicable across different institutions with their unique constraints, this system demonstrated solid results achieving around 93% effectiveness compared to manual scheduling or existing systems

dynamic chromosomes, which adapt their size according to the specific requirements of each department or institution. This approach allows the GAs to handle varying numbers of courses, ensuring that the algorithm remains flexible and can be applied across different academic settings. The use of dynamic chromosomes also helps in maintaining the diversity of the population, which is crucial for avoiding premature convergence to suboptimal solutions.

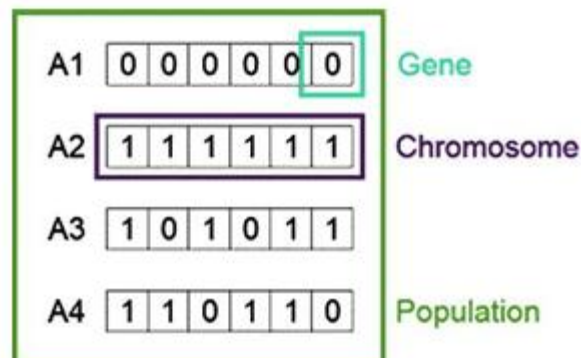


Figure 1.8: Chromosome Representation in GA [2]

As Key findings in it recognized that the dynamic chromosomes have a better impact on solving the university course scheduling problem , especially when compared to static chromosome approaches, in a static chromosome set up, the chromosome sized is fixed , which in return limits the algorithm's flexibility in handling varying numbers of courses or constraints within different departments ,which could lead to inefficiencies, furthermore lack of adaptability is shown to result in premature

convergence and provide suboptimal solutions, as the fixed structure may not adequately represent the diverse scheduling requirements

In contrast, dynamic chromosomes adjust their size based on the specific needs of each department or institution. This adaptability allows the genetic algorithm (GA) to more accurately represent and process the varying number of courses, leading to a more efficient exploration of the solution space.

The findings of this research highlight that dynamic chromosomes can reduce scheduling conflicts more effectively than their static counterparts. The ability to adapt the chromosome size directly correlates with a more accurate representation of real-world constraints, leading to a higher quality of generated timetables. This adaptability was a key factor in achieving the 93% effectiveness observed

Multi-Objective post enrolment course timetabling problem A new case Study [4]

This Research introduces a novel approach to address the complexity of real-world course timetabling issues. Unlike traditional single-objective problems, this research incorporates a new soft constraint designed to minimize the total number of waiting timeslots between courses for students throughout the day. By adding this constraint, the problem becomes more reflective of actual student needs, increasing both its complexity and its relevance to real-world scenarios.

In order to meet real-world scenarios, this research proposes a NSGA-II with a variable population size (VPS), which introduces a concept of given a lifetime for each individual in the population, which is evaluated at the time of its birth, this variable population size allows for more dynamic and adaptable search process, which is vital when dealing with the multi-objective environment of balancing several criteria.

The algorithm was tested on standard benchmark problems and results demonstrated significant improvements over the original NSGA-II with the introduction of the VPS mechanism

It's worth noting that when trying to satisfy both hard and soft constraints this research writes that it is impossible to satisfy all the soft constraints but in order to have a quality solution it is important to ensure that soft constraints are met as well

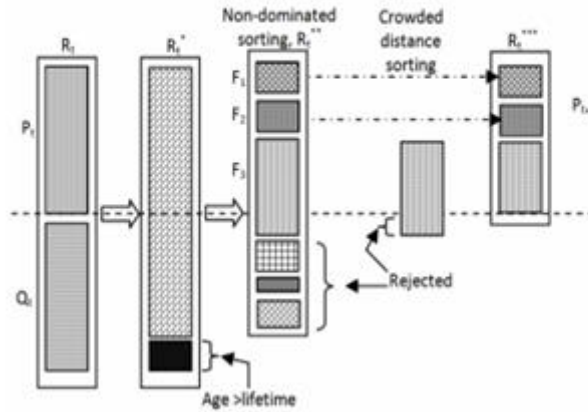


Figure 1.8: NSGA-II VPS Procedure [4]

Key Findings of the Research as follows

- **NSGA-II VPS Performance:** The proposed NSGA-II with Variable Population Size (VPS) outperformed the standard NSGA-II algorithm across various datasets, particularly in smaller datasets. The improvements were significant for small datasets and marginal for medium and large datasets.
- **Soft Constraint Violations:** NSGA-II VPS showed clear improvements in reducing violations of soft constraints
- **Optimization Strategy:** The use of a variable population size allowed the algorithm to better balance exploration and exploitation during the optimization process. This led to an enhanced ability to find high-quality solutions.
- **Lifetime Calculation:** The introduction of individual lifetimes based on solution quality ensured that lower-quality solutions were removed from the population pool, aiding the algorithm in maintaining a focus on promising areas of the solution space.
- **Comparison with Other Methods:** The NSGA-II VPS algorithm achieved competitive results when compared with well-known algorithms in the literature, particularly for larger datasets where it ranked second in performance.

Table 8. Comparison results between NSGA-II and NSGA-II VPS

Dataset	Approach	S1	S2	S3	S4 (New)	Total
<i>small4</i>	NSGA II	20	66	2	279	367
	NSGA II VPS	50	40	0	265	355
<i>medium3</i>	NSGA II	81	73	41	2978	3173
	NSGA II VPS	103	81	21	2933	3138
<i>large</i>	NSGA II	179	276	153	6071	6679
	NSGA II VPS	230	263	95	6027	6615

Figure 1.9: Comparison between NSGA-II vs NSGA-II VPS [4]

Future Work of Research:

- **Adaptive Crossover/Mutation Rates:** Future research will focus on intelligently controlling the crossover and mutation rates using fuzzy logic-based lifetimes. This aims to further enhance the balance between exploration and exploitation, potentially leading to even better optimization results.
- **Further Optimization:** Continued efforts will be made to improve the algorithm's performance, especially in reducing violations of specific soft constraints like S1, which did not show as much improvement.
- **Exploration of Additional Strategies:** The study suggests that additional strategies and variations of the NSGA-II VPS algorithm could be explored to enhance its performance further, particularly for larger and more complex datasets.

1.1.2 Background Survey

In Order to get a clear idea about the Current timetabling issues Background survey was conducted with Students, Academic Staff & Administrative Staff in Sri Lanka institute of information technology (SLIIT), The Extracted Results from Survey are listed below

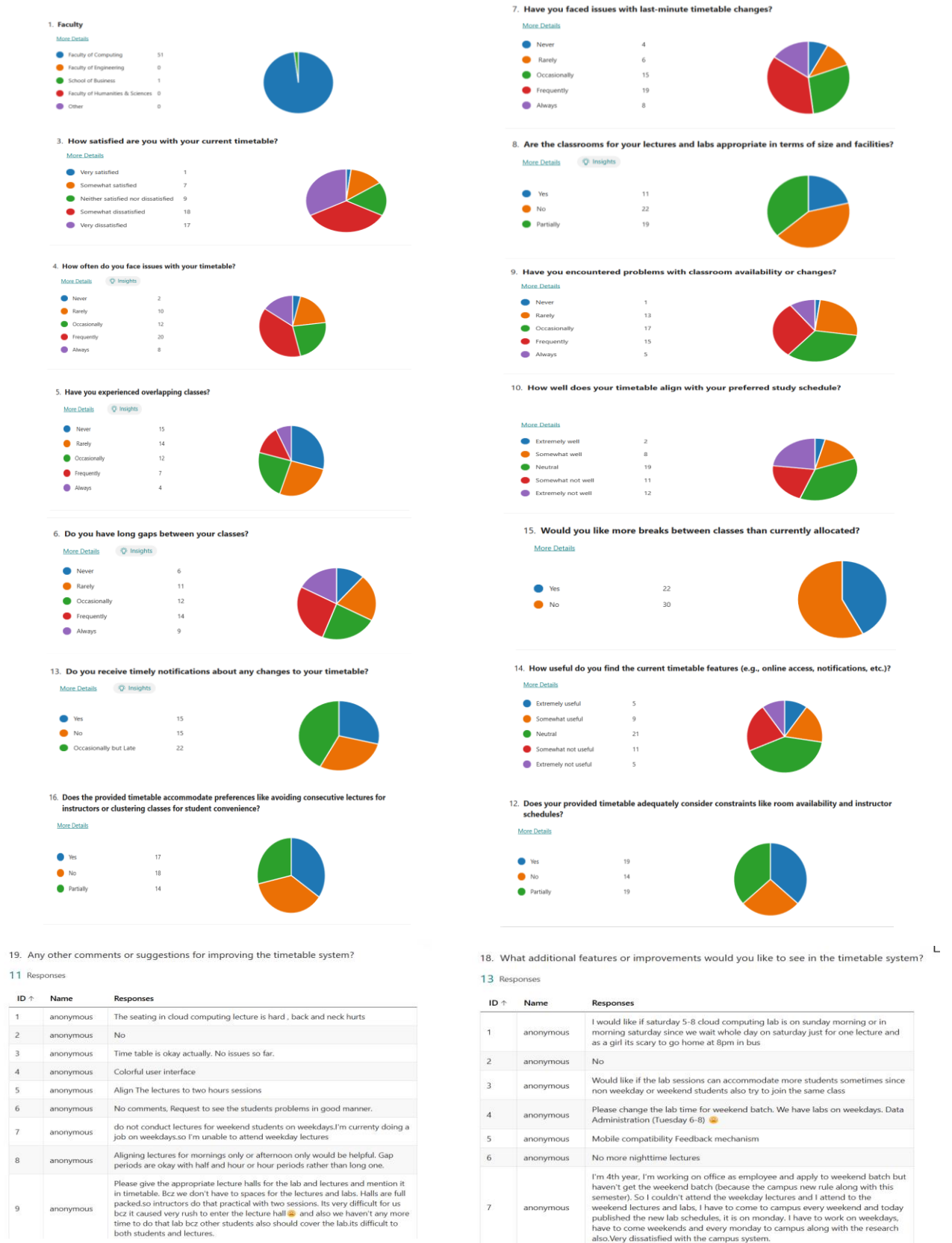


Figure 1.10: Student Background Survey Results

Overall Satisfaction: Many respondents expressed dissatisfaction with their current timetables, with many reporting that the schedules are only meeting somewhat of their needs

Timetable Issues: It has been noted that students frequently encounter issues with their timetables. These issues include but are not limited to overlapping classes, long gaps between lectures

Classroom Availability: It has been noted that in several instances, allocated student space was not enough to accommodate all the students that were enrolled in the certain course, which made it difficult for both students and academic staff to conduct and participate in sessions effectively

Alignment with preferred schedules: The survey revealed that the current timetable doesn't align well with the students' wished modes of studies, many students have expressed a preference for avoiding early morning or late evening classes, yet it has been noted that the timetables do not accommodate those inconveniences.

Notification and Communication: Timely communication about the timetable changes was a recurring issue, as there was no identified proper system for the timetable instead its always administration notification which are often known to come in late, which prompts dissatisfaction among users

Suggested improvements: Students provided several suggestions for improving the timetable system. Common themes included the desire for better alignment of lectures and labs to specific days, and proper notification system

1.1.3 Research Gap

As Shown in Table 1.1 Below Current body of research on university course timetabling, in the context of utilization of EAs, has provided substantial advancements in optimization techniques, however , significant gaps remain in fully addressing the diverse and complex constraints found in real-world university environments , while existing studies has implemented GAs and Variants of EAs such

as NSGA-II , the focus has been on improving algorithmic efficiency and solution quality, often in idealized scenarios with limited consideration of dynamic, and real time multi-faceted constraints

Furthermore, the conducted background survey revealed several issues that existing timetable solutions yet to solve adequately, these include but not limited to having large gaps between lectures, insufficient allocation, dynamic changes in real time and lack of timely notifications about timetable changes that take place

Despite various advancements in Evolution algorithm -based timetabling solutions[8] these issues indicate gap between the theoretical research and practical applications, which gives a focus on the need for further exploration into how EAs can be tailored to dynamically adapt to the evolving needs and preferences of every party involved in this process , this is not limited only to EA techniques but also integrating more flexible , user centric design principles into development of timetabling systems

This research aims to bridge this gap by developing a more robust and adaptable timetable solution that leverages the strengths of EAs While incorporating the complex, real world constraints identified in background survey

Table 1.1: Comparison of Existing System

Features	[1]	[2]	[3]	[4]	Component Proposed system: "TimeTable Wiz"
Usage of Clashes Method for Hard Constraints	✓	✓	✗	✓	✓
Genetic Algorithms using three parents.	✗	✓	✗	✗	✓
Dynamic Chromosomes	✗	✗	✓	✗	✓
Variable Population size	✗	✗	✗	✓	✓
Considerations of Multi objectives	✗	✗	✗	✓	✓
Integrated Into Fully Fledged Software	✗	✗	✗	✗	✓
Usage of Selection Method as Roulette wheel	✗	✗	✓	✓	✓
Usage Of Selection Method As Tournament	✓	✓	✗	✗	✓

1.1.4 Research Problem

University course scheduling is a complex problem [2] that involves various stakeholder preferences in the system, while trying to satisfy all the stakeholder constraints and preferences. Evolutionary algorithms have been widely applied to optimized timetabling solutions [8]

Despite Progress EA based solutions for timetable practical, implementations continue to fall short in addressing the diverse and dynamic needs of real-world university setting

In this scenario the key focus is on what manner GA along with MOEA Selected variants effectively adapt and integrate into timetable system to address complex, real-world constraints and user preferences, including dynamic changes, efficient allocation, timely notification, while finding an optimal timetable solution

Key aspects to address:

Dynamic Adaptation: developing EAs that can dynamically adjust in real time changes and evolving constraints that the timetable remains effective, which in case reduces the current systems cascading effect of one change affecting all others in the timetable [9]

Real-world-Constraints: incorporating real world multi objective constraints and user preferences to the EA-Based timetable approach

User-Centric Design: Role base access control system with use centric design principles to enhance the system's responsiveness to individual needs and preferences

Comprehensive Integration: Creating Software Solution which combines advanced EA Techniques with practical and real-world considerations

1.2 Research Objectives

The Primary Objective of this research is to design and develop advance timetable solutions which leverages Evolutionary algorithms and its variants such as GAs, NSG-II, SPEA2, MOEA/D and build a research framework upon the findings of these algorithms. Furthermore, this research aims not only to utilize these algorithms to address the inherent complexities and constraints of university timetabling, but also to build a comprehensive research framework to build upon a system, while facilitating the deeper exploration into the effectiveness of these algorithms in real world, dynamic environments. Main goal is to create a timetable system that is adaptable, efficient and a system which offers comprehensive software solutions to users

1.2.1 Specific Objectives

- Design Evolutionary Algorithm Frameworks Suitable for Scheduling:

Develop a tailored framework for EAs that is specifically optimized for the challenges of university course timetabling. This involves designing algorithmic structures that can efficiently navigate the complex space of scheduling, balancing multiple constraints and objectives to produce feasible and optimal timetables. The framework will be adaptable, allowing for the integration of various EA MOEA variants and fine-tuning of parameters to enhance performance across different scenarios.

Identify Key Areas of Improvement for the Algorithm: Conduct a thorough analysis of existing EA implementations in timetabling to pinpoint weaknesses and areas where enhancements can be made. This objective focuses on understanding the limitations of current algorithms in handling dynamic constraints, solution quality, and computational efficiency. By identifying these key areas, the research aims to inform the development of more robust and effective algorithms tailored to real-world timetable challenges.

Implement Variations of Evolutionary Algorithms Such as NSGA-II, MOEA/D, and SPEA2: Integrate and implement advanced variants of Evolutionary Algorithms, NSGA II (NSGA-II), Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), and Strength Pareto Evolutionary Algorithm 2 (SPEA2). These variations will be tested and compared to determine their effectiveness in

optimizing multiple conflicting objectives simultaneously, such as minimizing clashes and providing quality solutions which even consider soft constraints

- Develop Role-Based Access Control (RBAC) on the Software:

Implement a robust Role-Based Access Control (RBAC) system within the scheduling software to manage and secure access to the timetable system. This will ensure that different user roles, such as administrators, faculty, and students, have appropriate access levels, safeguarding sensitive data and enabling users to perform their specific tasks efficiently. RBAC will enhance the overall security and usability of the system.

Create Features to Allow Users to View and Export Their Perspective Timetables Design and integrate features that allow users to easily view and export their personalized timetables. This includes developing intuitive user interfaces where students and faculty can access their schedules, as well as tools for exporting timetables in various formats (e.g., PDF, CSV) for offline use or integration with other applications.

Integrate an API that Enables Users to Seamlessly Add Their Schedules to Personal or Institutional Calendar Applications: Develop and integrate an API that allows users to sync their timetables with personal or institutional calendar applications (e.g., Google Calendar, Microsoft Outlook). This API will provide seamless connectivity between the timetable system and external calendar tools, ensuring that users can easily manage their schedules and receive updates in real-time

2 SYSTEM METHODOLOGY

This chapter details the systematic approach employed for the design, development, and evaluation of the proposed university timetable scheduling system. The methodology integrates requirements engineering Software Overview and advanced optimization techniques, specifically focusing on the application of Multi-Objective Evolutionary Algorithms (MOEAs). by establishing the foundational requirements derived from stakeholder analysis and presenting the overall system architecture designed to meet these needs. Subsequently, the chapter delves into the technical core, explaining the data representation chosen for timetable solutions and the evolutionary algorithms (NSGA-II, MOEA/D, SPEA2) selected to navigate the complex optimization landscape, covering both their conceptual foundations and specific implementation details for the target datasets (See 2.5 for detailed dataset descriptions). A significant aspect of this work involves the development of specialized, domain-specific operators designed to enhance the performance of these algorithms, which are described in detail. Finally, the chapter concludes by outlining the experimental setup employed to rigorously evaluate the effectiveness of the proposed methodology. Requirement Analysis and Specification: This phase involves gathering requirements from the stakeholders of the systems such as students, administrative staff, and academic staff

2.1 Requirements

The development of the timetable scheduling system was guided by a set of requirements derived from analyzing the needs of key stakeholders, including students, academic staff (lecturers), and administrative personnel within a university context, particularly drawing parallels from environments like SLIIT. These requirements define the necessary functionalities, performance characteristics, and constraints the system must address.

2.1.1 Functional Requirements

Functional requirements specify the core operations the system must perform:

- **Algorithm Selection:** The system must allow authorized users (e.g., administrators) to choose from the available multi-objective evolutionary algorithms (NSGA-II, SPEA2, MOEA/D) for timetable generation.
- **Parameter Configuration:** Users should be able to configure key parameters for the selected algorithm, such as population size and number of generations, or use sensible default values.
- **Timetable Generation:** The system must initiate and execute the timetable generation process based on the selected algorithm, parameters, and the target dataset (e.g., SLIIT, Muni).
- **Constraint Handling:** The generation process must ensure that resulting timetables adhere to predefined hard constraints (e.g., no overlapping lectures for the same student group, lecturer, or room; respecting room capacity limits). It should also aim to optimize soft constraints (preferences).
- **Progress Monitoring:** Provide real-time feedback on the progress of the timetable generation process, potentially using technologies like Server-Sent Events (SSE).
- **Results Storage:** Generated timetable solutions and associated performance metrics must be persistently stored, for instance, in a MongoDB database.
- **Results Display and Access:** Allow authorized users to view the final optimized timetable results. Students and lecturers should be able to access the sections of the published timetable relevant to them.
- **State Management:** The system needs to maintain and display the state of generation tasks (e.g., pending, generating, completed, failed).

- **Publishing & Notification:** Administrators should be able to select and publish a generated timetable, triggering notifications or updates for relevant users.
- **Role-Based Access Control (RBAC):** Ensure users have appropriate permissions based on their roles (e.g., Admin can generate/publish, Lecturers/Students can view).

2.1.2 Non-Functional Requirements

- **Usability:** The user interface (particularly the React frontend) should be intuitive, providing clear feedback and requiring minimal training for users to perform their tasks.
- **Reliability:** The system must be robust, handling errors gracefully. Real-time updates (like SSE) should maintain stable connections. Data storage should be reliable (e.g., using MongoDB Atlas).
- **Performance:** The API endpoints (FastAPI) should be responsive. Real-time progress updates should be delivered in a timely manner without overwhelming the frontend or backend. Algorithm execution time should be reasonable for the problem size.
- **Maintainability:** The codebase should adhere to good software engineering practices and clear structure guidelines to facilitate future updates and modifications.
- **Scalability:** While the initial scope might be specific datasets, the architecture should consider potential future scaling to handle larger datasets or more concurrent users, particularly concerning backend processing and database interactions.

Security: The system must implement appropriate security measures to protect data and control access:

- **Authentication:** Securely verify the identity of users (students, lecturers, administrators) logging into the system.

- Authorization (RBAC): Enforce Role-Based Access Control to ensure users can only access data and perform actions appropriate to their role (e.g., only Admins can trigger generation or publish timetables; users can only view relevant schedule data).
- Data Protection: Protect stored data (timetables, user information) against unauthorized access or modification, both at rest (in the database) and in transit (e.g., using HTTPS for API communication).
- Input Validation: Protect against common web vulnerabilities (e.g., injection attacks) by validating all user inputs on the backend.
- Session Management: Implement secure session handling to prevent hijacking.

2.1.3 Use Case Diagram

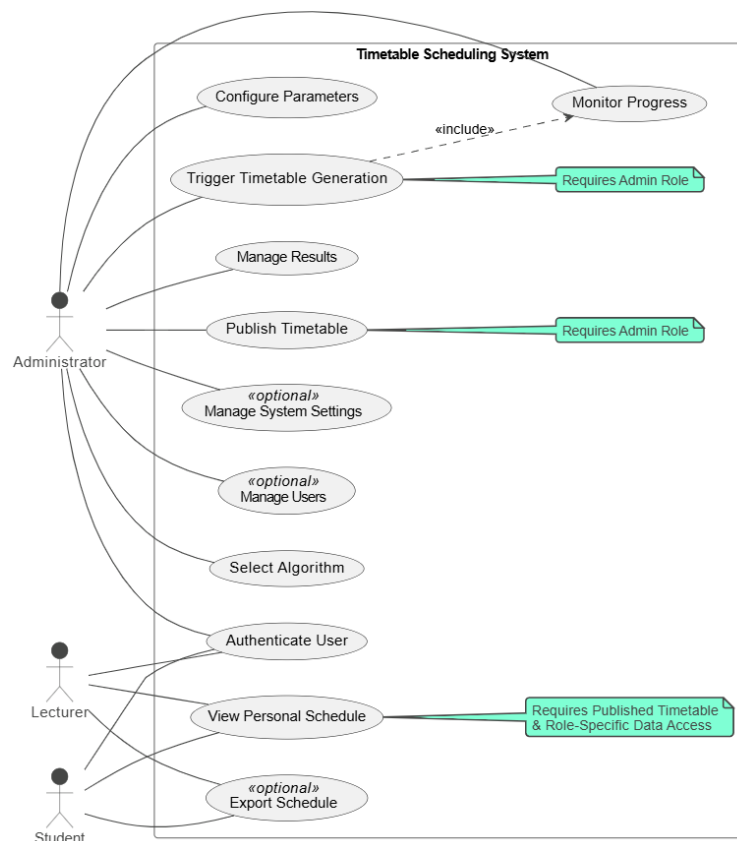


Figure 2.1: Use Case Diagram

2.2 System Overview

In Order to meet the Functional and Non Functional-Requirements proposed (Section 2.1) in the scheduling system integrates Evolutionary Algorithms (EAs) to generate optimized timetables, incorporating multiple evolutionary techniques such as NSGA-II, SPEA2, and MOEA/D along with other algorithm implementation. The system is designed to manage user requests through a Role-Based Access Control (RBAC) module, ensuring appropriate access levels and permissions. The architecture features a central API layer that communicates with database to handle data storage and retrieval. The Constraint Management Module applies predefined constraints to the scheduling process, The final timetable is managed by the Scheduling Module and can be synchronized with external calendar applications via the Calendar API Integration and users can view their relevant assigned timetables. This design ensures a robust, flexible, and user-centric scheduling solution.

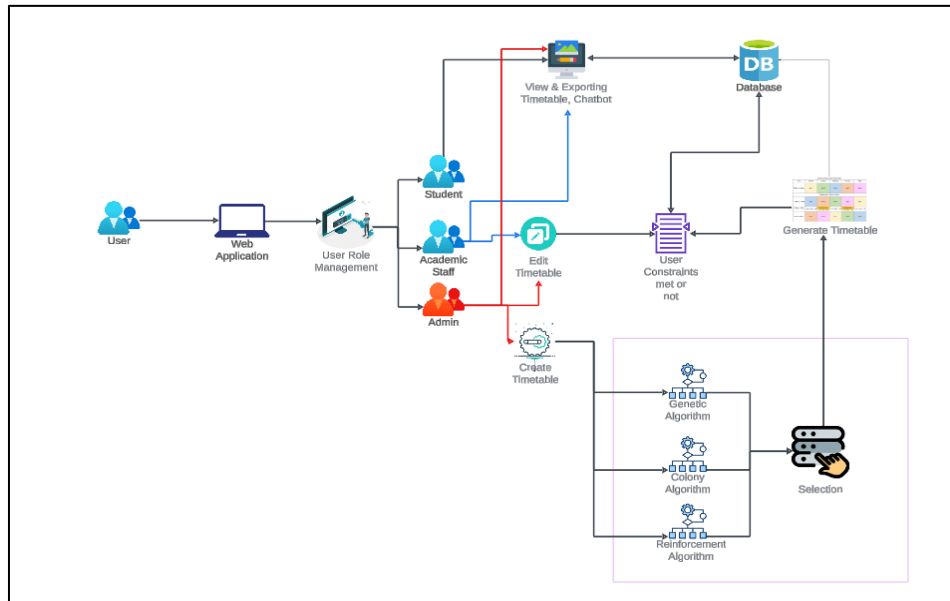


Figure 2.4: High-Level Diagram for overall system diagram

The main users of this application are students, Lecturers (Academics), and Administration. All users can access the implemented application through the web interface. Data are stored in the MongoDB Atlas database cluster to prevent data loss. From the application, an admin user can perform timetable generation using selected algorithms. Assuming they have domain knowledge, the administrator can set up the

number of generations and populations for the selected algorithms; if not, the timetable will be generated using default values provided on the server. After the timetable generation, the administrator can select and publish it. This published timetable will be recognized system-wide, and relevant users will be notified and updated. They can then view the new timetable through their own logins provided by the system. This process eliminates the identified problem found in environments like SLIIT, where there was no system-wide notification system to inform users about timetable updates and changes.

2.3 Solution Representation

The way a potential timetable is encoded (the chromosome structure) is crucial for EAs, Different Representations were used for two datasets due to their inherent structural differences

For the SLIIT dataset, a timetable is represented as a nested dictionary structure:

- ``schedule[slot][space] = activity_id``.
- The outer keys are the time slots (e.g., 'MON1', 'MON2', ..., 'FRI8').
- The inner keys are the available spaces (rooms/labs, e.g., 'LH401', 'LAB501').
- The value assigned is the ID of the `Activity` scheduled in that specific time slot and space, or `None` if the slot/space is free.

Each individual chromosome in the population directly corresponds to one such complete `schedule` dictionary.

For the Muni dataset (ITC format), a solution is represented using a dictionary within a `Solution` object: ``assignments[class_id] = (time_id, room_id)``.

- The keys are the unique IDs of the classes to be scheduled.
- The values are tuples, where `time_id` refers to a specific time pattern (combination of days, start time, length, weeks) and `room_id` refers to the assigned room.

Each chromosome represents a complete mapping of all classes to their time and room assignments. This direct mapping facilitates handling the complex constraints typical of the ITC format.

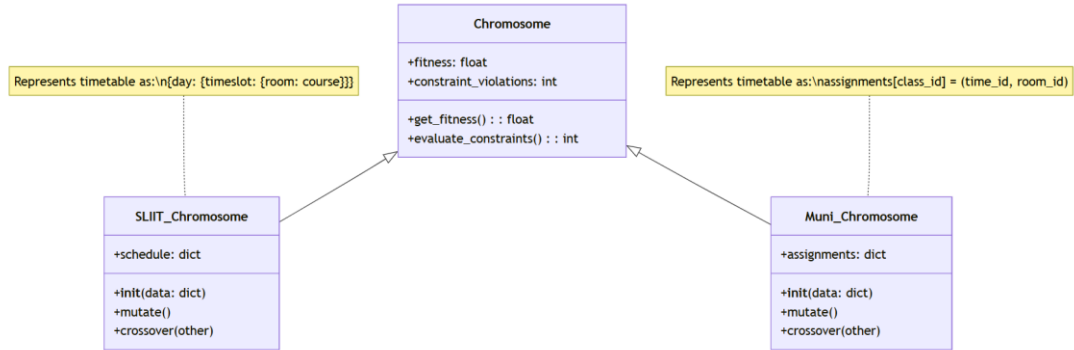


Figure 2.4: Chromosome Representation for Each Dataset

The rationale behind this representation can be structured this way for SLIIT dataset,

1. **Grid-like Structure** - Matches the physical timetable view (timeslots \times rooms) for intuitive visualization and debugging
2. **Direct Constraint Checking** - Enables efficient room conflict detection ($O(1)$ lookup for any slot/room)
3. **Simple Variation** - Swapping activities between timeslots/rooms maintains solution validity
4. **Dataset Characteristics** - Suits SLIIT's fixed weekly pattern with consistent room availability

For MUNI-FSPS-SPR17:

1. **Complex Time Patterns** - Tuples accommodate ITC's varied meeting patterns (different days/weeks/durations)
2. **Constraint Propagation** - Grouped assignments enable efficient checking of:
 - Teacher conflicts
3. **Compact Representation** - Scales better for large datasets ($O(n)$ vs $O(n \times m \times p)$ for SLIIT)
4. **Specialized Operators** - Enables domain-specific mutation/crossover that respects:
 - Room features
 - Time pattern constraints
 - Curriculum requirements

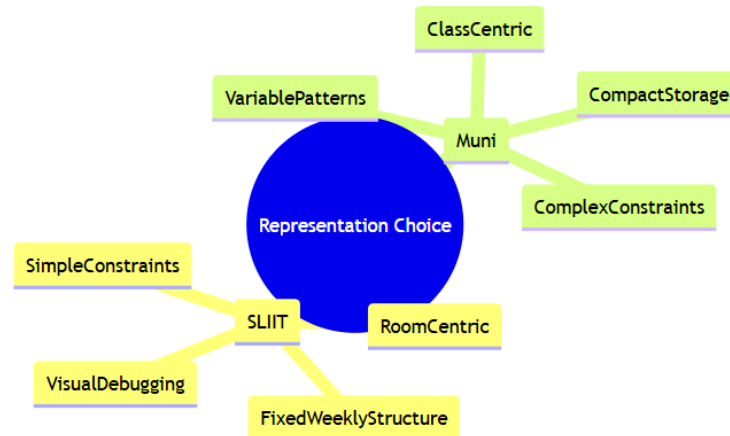


Figure 2.5: Chromosome Representation Choice

2.4 Algorithms

As Discussed Above the Research employs three established MOEAs adapted for the university timetabling problem NSGA-II , MOEA/D , SPEA2, under each algorithm specific implementation for both MUNI and SLIIT dataset is discussed after the common problem definition

2.4.1 Problem Definition

Let:

x be a **timetable solution** (a complete schedule assignment), where:

- For **Muni dataset**:

$$x = (ci, ti, ri)_{i=1}^n \quad n \times 3 = (ci, ti, ri)_{i=1}^n$$

where ci = class, ti = timeslot, ri = room

(Each class assigned to one (time, room) pair)

- For **SLIIT dataset**:

$$x = 3D \text{ tensor } [days] \times [timeslots] \times [rooms] [days] \\ \times [timeslots] \times [rooms]$$

(Matrix representation of weekly grid)

Then:

$Fitness(x) = (f_1(x), f_2(x))$, where:

$f_1(x)$ = Hard constraint violations

$f_2(x)$ = Soft constraint penalty score

The aim of the MOEAs is to find a set of non-dominated solutions representing the best possible trade-offs between minimizing $f_1(x)$ and maximizing the satisfaction of preference (represented by minimizing soft constraint penalty score $f_2(x)$)

2.4.2 Common Implementation Aspects:

Several implementation components were shared across the MOEAs employed:

- **Initialization:** The initial population (P_0) of candidate timetables was generated using methods tailored to dataset characteristics, as described in Section 2.3: constraint-aware sampling for the Muni dataset's assignment map representation, and grid distribution heuristics for the SLIIT dataset's schedule[slot][space] representation.
- **Fitness Evaluation:** All algorithms evaluate individuals x based on the hard ($f_1(x)$) and soft ($f_2(x)$) constraint objectives defined in Section 2.4.1. The specific constraint checking logic resides within a shared evaluation module, adapted for each dataset's rules.
- **Variation Operators (Standard):** Standard genetic operators were adapted for the specific solution representations (Section 2.3). This included:
 - *Crossover:* Problem-specific crossover operators (e.g., adapted uniform crossover) suitable for combining the dictionary/grid or assignment map structures.
 - *Mutation:* Operators applying small, random modifications (e.g., swapping activities, altering time/room assignments) based on a mutation probability, adapted for each representation.
- **Repair Mechanism (Muni Dataset):** For the Muni dataset, a repair function was available to be invoked post-variation to attempt correction of newly introduced hard constraint violations, guiding the search towards feasible regions.
- **Local Search (SLIIT Dataset):** For the SLIIT dataset, an optional periodic local search procedure could be applied to individuals. This involves exploring the immediate neighborhood of a solution (e.g., by swapping nearby activities) to potentially find improvements and accelerate convergence.

2.4.3 NSGA-II:

NSGA-II, developed by Deb et al. [12], is a widely used multi-objective evolutionary algorithm renowned for its efficiency and effectiveness in finding a well-distributed

set of non-dominated solutions. It improves upon earlier genetic algorithms by introducing several key mechanisms:

Fast Non-dominated Sorting: An efficient procedure ($O(MN^2)$ where M is objectives, N is population size) to rank solutions into different Pareto fronts based on dominance.

Crowding Distance: A density estimation metric calculated for solutions within the same front. It helps maintain diversity by favoring solutions in less crowded regions of the objective space during selection.

Elitism: Ensures that the best solutions found so far are preserved across generations. This is achieved by combining the parent and offspring populations before performing selection for the next generation based on rank and crowding distance.

Algorithm 01 NSGA-II:

```

function: NSGA-II-Timetabling ( $N, G$ )
input:  $N$ : integer – population size
          $G$ : integer – maximum number of generations
output:  $S$  – set of non-dominated timetable solutions

1   Initialize population  $P_0$  with  $N$  random timetables
2   Evaluate objectives  $f_1(x)$  and  $f_2(x)$  for each  $x \in P_0$ 
3   Apply fast non-dominated sorting to rank  $P_0$  into fronts  $F_1, F_2, \dots$ 
4   Calculate crowding distance for solutions in each front
5   for  $t = 0$  to  $G-1$  do
6       Select parents from  $P_t$  using binary tournament
7       Create offspring  $Q_t$  using crossover and mutation
8       Apply repair operator to maintain timetable validity
9       Evaluate objectives  $f_1(x)$  and  $f_2(x)$  for each  $x \in Q_t$ 
10       $R_t \leftarrow P_t \cup Q_t$ 
11      Apply fast non-dominated sorting to rank  $R_t$  into fronts
12       $P_{t+1} \leftarrow \emptyset$ ;  $i \leftarrow 1$ 
13      while  $|P_{t+1}| + |F_i| \leq N$  do
14           $P_{t+1} \leftarrow P_{t+1} \cup F_i$ ;  $i \leftarrow i + 1$ 
15      end-while
16      Sort  $F_i$  by crowding distance
17      Add solutions from  $F_i$  to fill  $P_{t+1}$  to size  $N$ 
18      Apply local search to selected solutions (periodically)
19      end-for
20      return non-dominated solutions from  $P_{t+1}$ 

```

Implementation: NSGA-II maintains a population of candidate timetables and iteratively refines them using selection, variation, and elitist environmental selection. Each generation, solutions are evaluated based on hard constraints (e.g., classroom and

teacher conflicts) and soft constraints (e.g., student preferences). The fast non-dominated sorting procedure ranks solutions into Pareto fronts according to dominance relationships:

For each solution p , two values are computed:

- (n_p) : the number of solutions dominating p
- (S_p) : the set of solutions dominated by p

Solutions with $(n_p = 0)$ form the first front. Subsequent fronts are constructed by decrementing the (n_p) value of solutions in (S_p) and adding those reaching $(n_p = 0)$ to the next front. This approach, originally proposed by Deb et al. [12], has been demonstrated to be particularly effective for handling the multi-faceted constraints.[19].

The crowding distance for solution i in front f is calculated as:

$$d_i = \sum_{m=1}^M \frac{f_m^{i+1} - f_m^{i-1}}{f_m^{max} - f_m^{min}}$$

Where (f_m^{i+1}) and (f_m^{i-1}) are the objective values of the next and previous solutions in the sorted front for objective m , and (f_m^{max}) and (f_m^{min}) are the maximum and minimum values for objective m . For timetabling specifically, this preserves diverse scheduling options—some favoring faculty constraints, others prioritizing student preferences.

Parent selection is performed using binary tournament selection, prioritizing lower-rank individuals and, in case of ties, those with greater crowding distance. Offspring are generated using problem-specific crossover and mutation operators adapted to the timetable representation, with a repair operator invoked as needed to correct hard constraint violations. After evaluating offspring, the parent and offspring populations are merged, and a new population is formed by filling up from the best fronts, using crowding distance to break ties when necessary.

This elitist selection process ensures that the best solutions discovered so far are preserved while maintaining diversity, which is crucial for exploring the wide range of possible timetables. Periodic local search can be applied to further refine selected solutions. The final output is a diverse set of non-dominated timetables, providing decision-makers with multiple feasible scheduling options that balance different priorities

2.4.4 MOEA/D:

MOEA/D, proposed by Zhang and Li [13], decomposes the M-objective problem into N scalar optimization subproblems using uniformly distributed weight vectors (λ). Key Characteristics as follows:

- **Decomposition Strategy:** Transforms the multi-objective problem into multiple single-objective optimization problems using scalarizing functions.
- **Neighborhood-Based Evolution:** Maintains a neighborhood structure based on weight vector similarity, focusing evolution within localized regions of the search space.
- **Efficient Resource Allocation:** Updates solutions based on improvement in scalarized values, concentrating computational effort on promising regions of the objective space.

Algorithm 02 MOEA/D

```

function: MOEA/D-Timetabling ( $N, G, T$ )
input:  $N$ : integer – population size
          $G$ : integer – maximum number of generations
          $T$ : integer – neighborhood size
output:  $EP$  – set of non-dominated timetable solutions

1   Generate  $N$  weight vectors and compute  $T$ -sized neighborhoods  $B(i)$ 
2   Initialize population  $P$  with random timetables and evaluate objectives
3   Initialize reference point  $z^*$  and external population  $EP = \emptyset$ 
4   for  $t = 0$  to  $G-1$  do
5       for  $i = 1$  to  $N$  do
6           Select parents from  $B(i)$  and generate offspring  $c$ 
7           Apply repair operator to  $c$  if needed
8           Evaluate objectives for  $c$  and update reference point  $z^*$ 
9           Update solutions in  $B(i)$  if  $c$  provides better aggregation values
10          Update  $EP$  with non-dominated solutions
11      end-for
12      Apply local search to selected solutions (periodically)
13  end-for
14  return  $EP$ 

```

Implementation For timetabling optimization, MOEA/D uses weight vectors to represent different preferences between hard constraints (classroom conflicts, teacher availability) and soft constraints (student preferences, department requirements). Each vector λ_i creates a unique subproblem, with neighborhoods $B(i)$ grouping similar subproblems together.

The Tchebycheff approach is used to scalarize the objectives for each subproblem:

$$g^{te}(x|\lambda_i, z^*) = \max \lambda_i^1 |f^1(x) - z^1|, \lambda_i^2 |f^2(x) - z^2|$$

Where $(\lambda_i^1, \lambda_i^2) = (\lambda_{i-1}^1, \lambda_{i-2}^1)$ is the weight vector and (z^*) is the current reference point containing the best values found for each objective. This scalarization focuses the search on minimizing the largest weighted deviation among objectives, which is particularly effective for exploring the "island" feasibility regions common in timetabling problems (i.e., feasible solutions are clustered in isolated regions of the search space, separated by large areas of infeasibility) [13].

Parent selection occurs within neighborhoods, allowing specialized constraint-handling strategies to emerge. For example, one neighborhood might excel at minimizing teacher conflicts while another focuses on room utilization. When a new timetable solution improves a subproblem, it can replace existing solutions in its neighborhood, focusing computational resources on promising scheduling patterns.

The external population (EP) maintains diverse non-dominated timetables discovered throughout the search. This provides multiple viable options representing different trade-offs, such as faculty preference versus student convenience.

2.4.5 SPEA2:

SPEA2 proposed by Zitzler, Laumanns, and Thiele [14], is another prominent MOEA that utilizes an external archive of a fixed size to explicitly store and maintain non-dominated solutions found during the search. Key features include:

- **Fine-Grained Fitness Assignment:** Calculates fitness based on both the strength of a solution (how many solutions it dominates) and its raw fitness (derived from the strengths of its dominators).
- **Density Estimation:** Incorporates a k-th nearest neighbor density estimation technique to differentiate between solutions with the same rank, promoting diversity.
- **Archive Management:** Uses an archive of fixed size (N' , often equal to the population size) and includes a truncation mechanism based on density to manage the archive when too many non-dominated solutions are found.

Algorithm 03 SPEA2:

```

function: SPEA2-Timetabling ( $N, \bar{N}, G$ )
input:  $N$ : integer – population size
       $\bar{N}$ : integer – archive size
       $G$ : integer – maximum number of generations
output:  $AG$  – set of non-dominated timetable solutions (final archive)

1   Initialize population  $P_0$  and empty archive  $A_0 = \emptyset$ 
2   for  $t = 0$  to  $G-1$  do
3       Calculate fitness  $F(i)$  for each solution in  $P_t \cup A_t$  based on:
4           - Dominance strength (how many solutions it dominates)
5           - Density estimation (distance to k-th nearest neighbor)
6       Form archive  $A_{t+1}$  with best solutions and adjust to size  $\bar{N}$ 
7       Select parents from  $A_{t+1}$  using binary tournament
8       Create offspring population  $Q_{t+1}$  using crossover and mutation
9       Apply problem-specific repair to offspring if needed
10      Evaluate objectives for  $Q_{t+1}$  and set  $P_{t+1} = Q_{t+1}$ 
11      Apply local search to selected archive solutions (optional)
12  end-for
13  return final archive  $AG$ 

```

Implementation: utilizes the common components detailed in Section 2.4.2 (Initialization, Fitness Evaluation, Variation, Repair, Local Search). Its unique aspects are the use of an external archive of fixed size to maintain elite non-dominated timetable solutions throughout the evolutionary process, and a fine-grained fitness assignment or timetabling problems, the SPEA2 fitness calculation involves several components:

The strength value $S(i)$ represents how many other timetables solution i dominates:

$$[S(i) = |j \mid j \in P_t \cup A_t \wedge i \succ j]$$

The raw fitness $R(i)$ aggregates the strengths of all solutions that dominate i , creating a gradient toward feasible regions in the timetabling search space:

$$\left[R(i) = \sum_{j \in P_t \cup A_t, j \succ i} S(j) \right]$$

The density estimation approach was proposed in the original spea2 paper [13] and has been shown effective for maintaining diversity in complex search spaces such as timetabling [14]

$$k = \left(\sqrt{|P_t| + |A_t|} \right) : \left[D(i) = \frac{1}{\sigma_i^k + 2} \right]$$

where (σ_i^k) is the distance to the k -th nearest neighbor. The final fitness combines both components: $[F(i) = R(i) + D(i)]$

At each generation, all individuals from the current population and the archive are evaluated, and their fitness is calculated using this scheme. The archive is then updated to include the best non-dominated solutions, applying a truncation mechanism based on density estimation if the archive exceeds its size limit. Studies on timetabling problems have shown that an archive size of 1.5 times the population size provides a good balance between solution diversity and computational efficiency.

Truncation iteratively removes the solution with minimum distance to its neighbors, preserving boundary solutions that represent important trade-offs in timetabling. This preservation is particularly valuable in the university timetable, where extreme solutions often represent critical trade-offs between faculty preferences and institutional constraints .

Parent selection for generating new timetables is performed exclusively from the archive using binary tournament selection. Offspring are created using problem-specific crossover and mutation operators, with a repair operator invoked as needed to

address hard constraint violations. The new population is then evaluated, and the process repeats.

This archive-driven approach ensures that high-quality and diverse timetable solutions are preserved, while the density-based truncation mechanism prevents loss of diversity among elite solutions. The final archive provides a set of non-dominated timetables, offering decision-makers a variety of feasible and high-quality scheduling options that balance different institutional priorities. Previous comparisons of evolutionary algorithms for timetabling have demonstrated that archive-based approaches like SPEA2 provide superior diversity among final solutions, a critical factor when university administrators need to evaluate multiple scheduling alternatives.

2.5 Experimental Setup:

The following subsections outline the datasets, algorithm settings, evaluation metrics, experimental procedures, and key aspects of the system’s UI/UX.

2.5.1 Dataset Descriptions:

Muni Timetabling Dataset (muni-fsps-spr17, ITC 2019 Instance):

The Muni-fsps-spr17 dataset is a real-world university timetabling instance originating from the International Timetabling Competition 2019 (ITC 2019). It comprises:

- **Courses and Classes:**
226 courses and 561 classes, with 191 fixed classes (classes with only one feasible placement).
- **Rooms:**
44 rooms of varying capacities and types.
- **Constraints:**
331 hard and 69 soft distribution constraints, with an average soft penalty of 1.22. There are 2,284 hard class pairs and 851 soft class pairs, with an average pair penalty of 1.42.
- **Students:**
865 students, each enrolled in an average of 7.76 courses and 11.60 classes.

- **Domains:**
Classes have an average of 20.24 feasible times and 3.15 feasible rooms, with an average domain size of 55.61 and 92% average availability for class placements.
- **Temporal Structure:**
The timetable spans 19 weeks, with classes averaging 90.97 minutes per meeting, typically meeting once per week for 9.3 weeks.
- **Utilization:**
Each class is taught for an average of 731.6 minutes per semester, rooms are occupied for 6,008.9 minutes, and students spend 7,728.2 minutes in class on average.
- **Constraint Weights:**
Time, room, distribution, and student-related objectives are weighted at 25, 1, 15, and 100, respectively.

As part of ITC 2019, this dataset presents a highly constrained and realistic scheduling environment, making it a standard benchmark for evaluating advanced timetabling algorithms [18].

FCSC Faculty of Computing Timetabling Dataset:

This dataset represents a university timetabling instance for the Faculty of Computing (FCSC) Simulated to Test the Algorithms. It comprises:

- **Modules and Activities (Courses and Classes):**
 - **30** distinct modules (courses).
 - **195** distinct activities (classes) that need scheduling, broken down into **30** Lectures, **150** Tutorials, and **15** Practical sessions.
- **Rooms (Spaces):**
 - **6** unique rooms available for scheduling.
 - These include **3** Lecture Halls (capacity **200** each) and **3** Computer Labs (capacity **60** each).
 - All **6** rooms are equipped with projectors and air conditioning; the **3** labs have computers.

- **Students and Groups:**
 - Represents a student body divided into **40** distinct subgroups across 4 years and 2 semesters.
 - Each subgroup contains **40** students, representing a total of **1600** students.
 - **49** lecturers are listed.
- **Domains:**
 - Activities have specified room type requirements (Lecture Hall, Tutorial Room, Computer Lab).
- **Temporal Structure:**
 - Activities have durations of **1** hour (Tutorials) or **2** hours (Lectures, Practicals)
- **Utilization:**
 - The total scheduled contact time across all activities is **240** hours per cycle (weekly).

Key Characteristics in this dataset is to test under the extreme capacity constraint and managing realistic student count across the timetable

2.5.2 Algorithm Settings:

Algorithm configurations were standardized where possible to ensure fair comparisons and to stay within the hardware limitations, while respecting algorithm-specific requirements. Each algorithm was configured as follows:

Common Parameters:

- Population size: 100 individuals
- Maximum generations: 200
- Crossover rate: 0.9 (two-point crossover)
- Mutation rate: 0.1 (random resetting mutation)

- Selection mechanism: Binary tournament selection
- Timetable representation: Direct encoding with activity-timeslot-room assignments
- Hard constraint penalty factor: 1000 (to guide the search away from infeasible regions)

Algorithm-Specific Parameters:

- **NSGA-II:**
 - Crowding distance threshold: 0.1
 - Ranking scheme: Fast non-dominated sorting with $O(MN^2)$ complexity
 - Diversity preservation: Crowding distance operator in objective space
- **MOEA/D:**
 - Decomposition approach: Tchebycheff
 - Weight vectors: 100 uniformly distributed vectors
 - Neighborhood size: 20 ($T = 20\%$ of population)
 - Neighborhood selection probability: 0.9
 - Maximum number of replacements: 2
- **SPEA2:**
 - Archive size: 100 (equal to population size)
 - K value for density estimation: 10 (\sqrt{N} as recommended in the literature)
 - Environmental selection: Truncation based on fitness and diversity

Evaluation Metrics

The performance evaluation employed multiple metrics to assess different aspects of the timetable solutions:

Feasibility Metrics:

- Hard Constraint Violations (HCV): Count of violations across all hard constraints
- Hard Constraint Satisfaction Rate (HCSR): Percentage of constraints successfully satisfied
- Constraint-specific rates: Room capacity satisfaction, teacher conflict avoidance, etc.

Quality Metrics:

- Soft Constraint Penalty (SCP): Weighted sum of soft constraint violations
- Objective-specific penalties: Student preferences, teacher preferences, room utilization
- Utilization Efficiency: Percentage of available timeslots and rooms effectively used

Multi-objective Performance Indicators:

- Hypervolume (HV): Volume of objective space dominated by the Pareto front
- Spread (Δ): Distribution of solutions along the Pareto front
- Inverted Generational Distance (IGD): Proximity to the true Pareto front
- Execution Time: Computational time required to generate solutions

Experimental Procedures:

The experimental procedure was designed to ensure statistical validity and reproducibility:

Execution Protocol:

- Each algorithm was executed on both datasets (SLIIT and Muni)
- Experiments were conducted on identical hardware configuration

Analysis Methodology:

- Results were aggregated using mean and standard deviation
- Statistical significance was assessed using the Wilcoxon signed-rank test ($\alpha = 0.05$)
- Effect size was calculated using Cliff's delta

Validation Approach:

- Cross-validation on historical timetables
- UX Generated Timetable
- Cross Check with Evaluation Method in Research

2.5.3 System UI/UX and Usability:**Algorithm Interaction and Configuration**

The algorithm interaction components prioritize usability while maintaining advanced functionality:

Administrators initiate timetable generation; the workflow guides them through algorithm selection and configuration. The interface presents algorithms via a dropdown menu. Upon selecting an algorithm, the system displays a concise description highlighting its strengths and appropriate use cases. Below this description, parameter fields appear with pre-populated default values suitable for most scenarios.

For example, selecting NSGA-II reveals' fields for population size and number of generations with recommended values of 100 and 50 respectively. These fields include validation to prevent invalid inputs, with visual feedback appearing for out-of-range values. Throughout this process, the interface maintains a clean, uncluttered appearance despite the complexity of the underlying algorithms, making advanced optimization accessible to non-technical administrators.

After configuring parameters, administrators click the prominent "Generate" button, initiating the timetable creation process. The system provides visual feedback during computation, and upon completion, Timetable can be viewed and then can be selected and published system wide. The export feature allows to view timetable in the HTML Format

Generate New SLIIT Timetable

* Timetable Name

e.g., SLIIT Summer 2025 Timetable

* Algorithm

NSGA-II (Non-dominated Sorting Genetic Algorithm II)

NSGA-II (Non-dominated Sorting Genetic Algorithm II)

SPEA2 (Strength Pareto Evolutionary Algorithm 2)

MOEA/D (Multi-objective Evolutionary Algorithm Based on Decomposition)

DQN (Deep Q-Network)

SARSA (State-Action-Reward-State-Action)

Implicit Q-learning

NSGA-II is a multi-objective optimization algorithm that uses a non-dominated sorting approach. It excels at finding a diverse set of Pareto-optimal solutions, making it effective for complex timetabling problems with competing objectives.

Generate New SLIIT Timetable

* Timetable Name

e.g., SLIIT Summer 2025 Timetable

* Algorithm

NSGA-II (Non-dominated Sorting Genetic Algorithm II)

* Population Size

100

* Number of Generations

50

Cancel

Generate

Algorithm Information

NSGA-II is a multi-objective optimization algorithm that uses a non-dominated sorting approach. It excels at finding a diverse set of Pareto-optimal solutions, making it effective for complex timetabling problems with competing objectives.

Timetable (Genetic algorithms)

Selected

Publish Timetable

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 Semester 1

Periods	Monday	Tuesday	Wednesday	Thursday	Friday
08.30 - 09.29	CS211 (Lab 06)	-	-	-	-
09.30 - 10.29	CS202 (LH 16)	-	-	-	-
10.30 - 11.29	CS202 (LH 16)	CS201 (LH 19)	-	-	-
11.30 - 12.29	-	CS201 (LH 19)	-	-	-
12.30 - 13.29	-	-	-	-	-
13.30 - 14.29	CS202 (LH 16)	CS112 (Lab 07)	CS202 (LH 10)	CS203 (LH 13)	-
14.30 - 15.29	-	CS112 (Lab 07)	CS202 (LH 10)	CS203 (LH 13)	-
15.30 - 16.29	-	-	-	-	-
16.30 - 17.29	-	-	-	-	-

Details for Wednesday

Subject: Object-Oriented Programming

Room: Lecture Hall 10 (LH010)

Teacher: Tharushi Rajapakse

Duration: 2 hours

Figure 3.3: Timetable Generation via Selected Algorithms

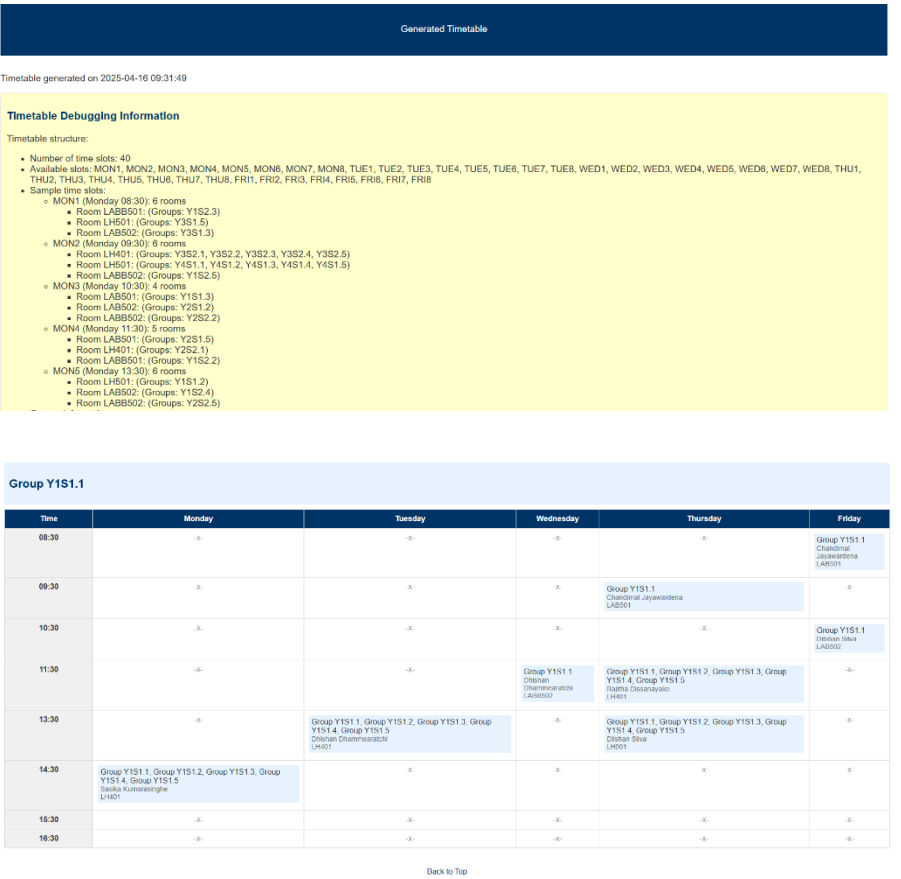
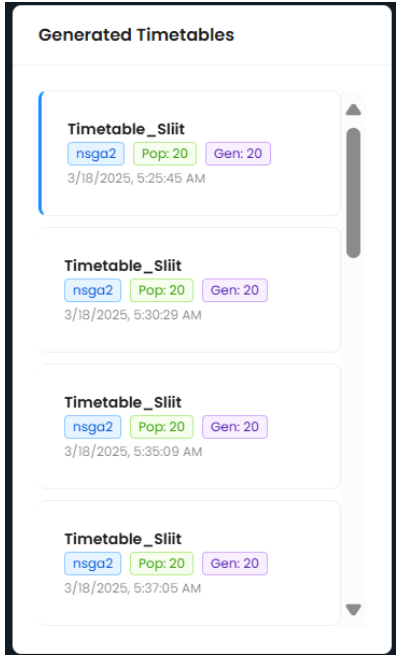
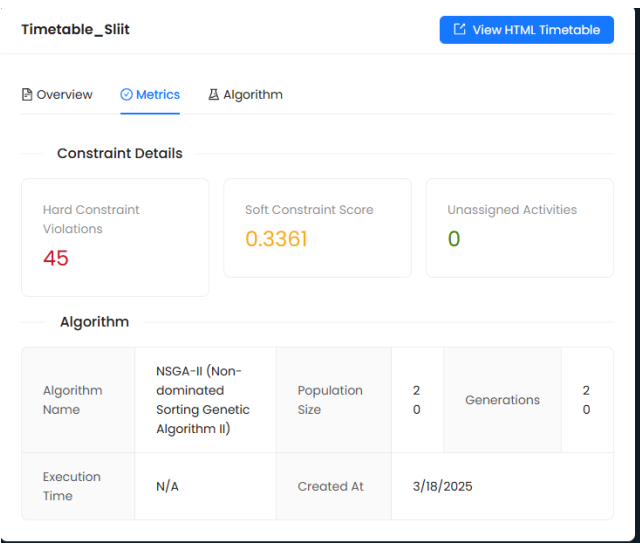
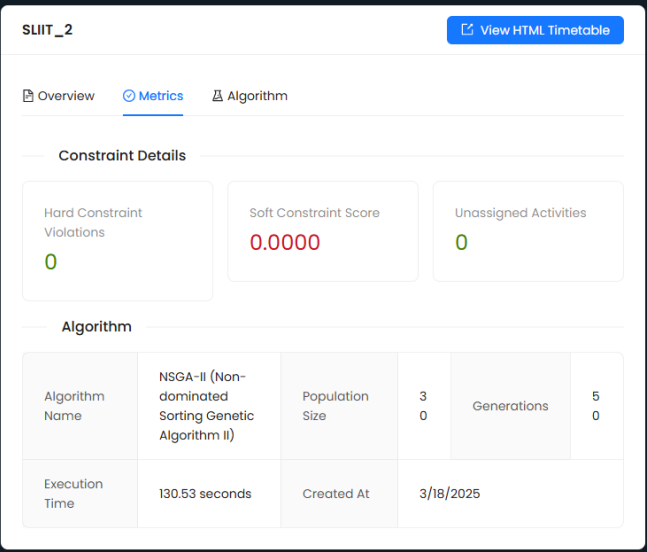


Figure 3.4: Timetable Results via Selected Algorithms and Exported HTML format

RBAC:

Upon accessing the system, users are directed to role-appropriate interfaces that shape their entire experience. Students entering the platform immediately see their personalized dashboard with a clean weekly view of their current semester's schedule. The interface presents a familiar calendar-like grid showing courses across weekdays and time slots, with each course rendered as a distinct card displaying essential information. Students can easily toggle between the weekly timetable view and a consolidated subjects list for a different perspective on their schedule. When selecting a specific course, a detailed popup reveals comprehensive information including the instructor, room details, and duration.

Faculty members experience a different journey, beginning with their teaching schedule organized by semester. Their interface emphasizes the courses they teach rather than those they attend, with clear indications of student groups assigned to each session. Faculty can navigate between semesters using a simple selector at the top of their dashboard. The system also provides teachers with an availability management calendar where they can mark dates when they are unavailable, helping administrators avoid scheduling conflicts. Each scheduled session displays the assigned room and student group information directly within the interface, eliminating the need for additional navigation.

Administrators have access to the most comprehensive interface, starting with the timetable generation dashboard. From here, they can view previously generated timetables or initiate the creation of new ones. The "Generate New Timetable" button opens a modal where administrators name the timetable and select the optimization algorithm to be used. This modal offers contextual information about each algorithm, helping administrators make informed selections even without deep technical knowledge of evolutionary algorithms.(described in above Section)

TimeTableWiz

Dashboard

Logout

Student Dashboard

My Class Schedule

Current Semester: SEM202

[Weekly Timetable](#)
[Subjects List](#)

Period	Monday	Tuesday	Wednesday	Thursday	Friday
08.30 - 09.29	-	-	-	-	SoftwareEng Room: Lecture Hall 06
09.30 - 10.29	-	-	Completet Room: Lecture Hall 12	-	SoftwareEng Room: Lecture Hall 06
10.30 - 11.29	-	-	Completet Room: Lecture Hall 12	-	BigData Room: Laboratory 04
11.30 - 12.29	-	-	-	-	BigData Room: Laboratory 04
12.30 - 13.29	-	-	-	-	-
13.30 - 14.29	Completet Room: Lecture Hall 12	IoT Room: Lecture Hall 13	InfoRetrieval Room: Lecture Hall 20	-	-
14.30 - 15.29	Completet Room: Lecture Hall 12	IoT Room: Lecture Hall 13	InfoRetrieval Room: Lecture Hall 20	-	-
15.30 - 16.29	-	-	-	-	-

My Teaching Schedule

Current Faculty ID: FA00000004

[All Semesters](#)
[SEM101](#)

SEM101

Period	Monday	Tuesday	Wednesday	Thursday	Friday
08.30 - 09.29	-	-	NumMeth Room: Lecture Hall 03	-	-
09.30 - 10.29	-	-	NumMeth Room: Lecture Hall 03	-	Compllog Room: Lecture Hall 19
10.30 - 11.29	-	-	-	-	Compllog Room: Lecture Hall 19
11.30 - 12.29	-	-	-	-	-
12.30 - 13.29	-	-	-	-	-
13.30 - 14.29	-	-	-	-	-
14.30 - 15.29	-	-	-	-	-
15.30 - 16.29	-	-	-	-	-
16.30 - 17.29	-	-	-	-	-

Manage Availability

Use the calendar below to mark days when you are unavailable to teach.

Note: Weekends are already marked as non-working days.

2025

Apr

Month

Year

Su

Mo

Th

Fr

Sa

Weekend

Weekend

Weekend

Weekend

30

31

01

02

03

04

05

06

07

08

09

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

Unavailability

Mark Unavailability

Date: 2025-04-17

Reason for unavailability:

Enter reason for unavailability

Mark as Unavailable

Cancel

2.5.4 System Requirements

Software Requirements:

- In user-end – Web Browser Accessible Device
- In Development End - Suitable IDE
 - Npm/React installed
 - Pythonv3.9
 - Mongodb

Used Hardware Specification:

- 16 GB Ram
- Intel ® CORE™ i5-8350 CPU @ 1.70GHz

2.5.5 Challenges

During the Research these Challenges were Identified as below

Hardware Resource Limitations: Development and testing were conducted on a system with limited resources (16GB RAM and an Intel i5 processor), which significantly constrained the scale of algorithm execution. Population sizes and generation counts had to be carefully optimized, as larger values would lead to memory overflow or excessive execution times. This limitation required implementing memory-efficient data structures and introducing incremental evaluation techniques that reduced computational overhead

Algorithm Performance and Scalability: The computational complexity of multi-objective evolutionary algorithms posed a significant challenge when handling large university datasets. Even moderate-sized problems with hundreds of activities and constraints resulted in extensive computation times. We addressed this by optimizing the fitness evaluation functions,

Data Acquisition and Preprocessing: Obtaining comprehensive, consistent datasets from university systems proved challenging. The SLIIT dataset required extensive

cleaning and normalization to resolve inconsistencies in course codes, room designations, and faculty assignments. Similarly, adapting the ITC 2019 Muni dataset to our system's format required developing custom data loading functionality

Managing Complex Constraints: University timetabling involves numerous hard and soft constraints that often conflict with each other. Modeling these constraints in a computationally efficient manner while ensuring they accurately reflected institutional requirements required multiple iterations and consultation with domain experts. We developed a flexible constraint representation system that allowed for dynamic weighting and priority adjustments.

Evaluation Methodology: Developing objective metrics to compare algorithm performance across different datasets and constraint configurations required careful consideration. ultimately implemented a multi-faceted evaluation framework with weighted constraint satisfaction scores, computational efficiency metrics, and user satisfaction ratings.

Integration with Existing Systems: Ensuring our system could coexist with and import/export data from universities' existing student information systems and course management platforms required developing flexible API interfaces and data transformation services, adding complexity to the implementation.

2.6 Work Breakdown Structure

The workload of the complete research project is shown in Appendix C.

2.7 Gantt Chart

The Gantt chart of the complete research project is shown in Appendix B.

2.8 Commercialization aspects of the project

The timetabling optimization system addresses a significant need in Sri Lankan higher education institutions, where scheduling challenges directly impact resource utilization, student experience, and faculty satisfaction. Developed initially to address the specific requirements of Sri Lankan universities, the system offers a path to improved operational efficiency and stakeholder satisfaction through a phased commercialization approach.

Sri Lankan Market Analysis

With over 15 state universities and approximately 50 private higher education institutions in Sri Lanka, there exists a substantial local market opportunity for an advanced timetabling solution:

- **Current State:** Most Sri Lankan universities rely on either manual scheduling processes or basic spreadsheet-based systems
- **Pain Points:** Administrative staff typically spend 4-6 weeks per semester on timetable creation, with frequent adjustments needed
- **Market Readiness:** Recent initiatives by the Ministry of Education to digitize university operations create favorable conditions for technology adoption

Local Subscription Model

The system will be offered as a Software-as-a-Service (SaaS) solution with pricing tailored to the Sri Lankan market can be viewed in below table:

Plan	Features	Price (USD)	Price (LKR)
Basic	Timetable Generation Support for up to 200 courses 5 optimization runs per day Basic reporting	7.62	1500.00
Professional	All Basic Support for up to 500 Unlimited optimization Runs	15.24	3000.00
Enterprise	All Professional Unlimited Custom Dedicated instance24/7 support	20.00	6000.00
Research	Open Source Research project Features	Free	Free

Table 3.4: Subscription plan of 'TimeTableWiz Application' application

Special considerations for local institutions:

- State universities: 30% discount on all plans
- Multi-campus deployment: Additional 15% discount
- Implementation and training: Included free for first year
- Offline deployment option for institutions with Servers

The following represents the estimated monthly operational costs for the initial Sri Lankan market:

Component	Amount(USD)	Amount(LKR)
Variable cost		
Traveling	17	5000
Server charges Azure/Aws	100	30,0000
Internet charges	7	2,000.00
Total	124	37000

Table 3.5: Budget Plan per Month

Local Go-to-Market Strategy

Our initial commercialization strategy focuses on building strong relationships with key Sri Lankan institutions:

1. Pilot Implementations: Free pilot programs with SLIIT and 1-2 state universities to generate local case studies demonstrating:
 - Reduction in scheduling conflicts
 - Improvement in room utilization
 - Time savings for administrative staff
2. University Grants Commission (UGC) Engagement: Presenting the solution to UGC to explore potential for system-wide adoption across state universities
3. Partnerships with Local IT Service Providers: Establishing implementation partnerships with local IT firms that already service the higher education sector|

initially focusing on building a strong presence in the Sri Lankan market, refine the system based on local feedback before pursuing broader global opportunities, ensuring a sustainable growth trajectory with manageable operational cost

2.9 Testing and Implementation

The implemented timetabling optimization system underwent comprehensive testing to ensure both algorithmic correctness and practical usability. Testing is a critical phase to verify that the implemented solution satisfies the requirements and performs as expected in real-world educational scheduling environments. Test cases were designed to cover all functional and non-functional requirements, spanning algorithm performance, user interface operation, and system integration aspects.

Table 3.6: TimetableWiz test case 1: NSGA-II for SLIIT dataset

Test Case ID	TTW001
Test Case Scenario	NSGA-II optimization for SLIIT Computing dataset
Test Input Data	SLIIT dataset with 60 courses, 7 rooms, 50 student groups
Test Procedure	<ol style="list-style-type: none">1. Configure NSGA-II with population size of 1002. Set the number of generations to 503. Set crossover rate to 0.9 and mutation rate to 0.14. Execute algorithms on the dataset5. Measure hard constraint violations6. Measure soft constraint satisfaction7. Record execution time
Expected Outcome	Algorithm should produce timetable with zero hard constraint violations and soft constraint score within reasonable timeframe (1-5min)
Actual Outcome	Zero hard constraint violations achieved with soft constraint score of 0.33 in 267 seconds
Test Result	Pass

Table 3.7: TimetableWiz test case 2: for RBAC

Test Case ID	TTW0002
Test Case Scenario	Verification of role-specific permissions and interface elements
Test Input Data	User accounts with three different roles: Administrator, Faculty, Student
Test Procedure	<ol style="list-style-type: none"> 1. Login as Administrator and verify access to algorithm selection, parameter configuration, and timetable publishing 2. Login as Faculty and verify access to teaching schedule, availability management, limited to assigned courses 3. Login as Student and verify access to personalized class schedule with no administrative controls 4. Attempt to access unauthorized features across all roles
Expected Outcome	Each role should only access authorized features with appropriate interface elements
Actual Outcome	All roles correctly restricted to authorized features; unauthorized access attempts blocked
Test Result	Pass

Table 3.8: TimetableWiz test case 3: for System Integration Timetable Views.

Test Case ID	TTW0003
Test Case Scenario	User configuration (NSGA-II, Population: 50, Generations: 30)
Test Input Data	User-written digit
Test Procedure	<ol style="list-style-type: none"> 1. Administrator selects algorithms and parameters 2. System processes request and generates timetable 3. Administrator publishes timetable

	4. Faculty and students view role-appropriate schedule views
Expected Outcome	Complete workflow with proper data flow and role-specific visualization
Actual Outcome	Successful generation and visualization with correct data shown to each role
Test Result	Pass

Table 3.9: TimetableWiz test case 4: for faculty unavailability test case

Test Case ID	TTW0004
Test Case Scenario	Faculty unavailability workflow and timetable generation
Test Input Data	Faculty account with ID FA0000004, calendar dates 2025-04-11
Test Procedure	<ol style="list-style-type: none"> 1. Login as faculty member 2. Navigate to "Manage Availability" interface 3. Mark April 17, 2025, as unavailable with reason "Conference attendance" 4. Submit unavailability 5. System identifies scheduled classes for FA0000004 on April 17 6. Log out and login as administrator 7. Receive notification about unavailable faculty member 8. View list of qualified substitute faculty members for affected courses 9. Select substitute faculty (FA0000002) for the affected classes 10. Confirm substitution assignment 11. Verify updated timetable for both faculty members

Expected Outcome	1.Faculty unavailability should be successfully recorded 2. Administrator should receive notification of affected classes 3.System should suggest qualified substitute faculty based on department and expertise 4.Selected substitute faculty should be assigned to affected classes 5. Both original and substitute faculty should see appropriate schedule updates
Actual Outcome	1. Unavailability successfully recorded in system 2. Administrator notified of 1 affected class requiring substitution 3. System suggested 4 qualified faculty members from same department 4. Substitute faculty successfully assigned affected class.
Test Result	Pass

2.9.1 Functional Testing

- **Unit Testing** was performed on individual components, including algorithm operations (selection, crossover, mutation), constraint evaluation functions, API endpoints, and UI components. All components achieved test coverage.
- **Integration Testing** verified interactions between system modules, with particular attention to data flow between the algorithm engine, API layer, database, and user interface components.
- **Role-Based Access Control Testing** specifically validated that the three user roles (Administrator, Faculty, Student) had appropriate access restrictions: and

- Administrators could access all features, including algorithm selection, parameter configuration, timetable generation, and publishing controls
 - Faculty members could view their teaching schedules, set availability preferences, and access detailed information about their assigned courses and rooms
 - Students could only view their personalized class schedules and general timetable information
- **System Testing** validated the complete application through end-to-end workflows across all user roles, ensuring that timetables generated by administrators were correctly displayed to relevant faculty and students.

2.9.2 Non-Functional Testing

- Usability Testing was conducted with representative users from each role group. Administrators, faculty, and students were asked to perform role-specific tasks without prior training.
- Performance Testing measured system responsiveness under various load conditions. The system maintained acceptable performance (API response under 500ms) on modest hardware (16GB RAM, Intel i5 processor).
- Security Testing focused on role-based access control mechanisms, authentication processes, and data protection. All attempts to access unauthorized functions were successfully blocked, and proper session management was verified.

3 Results & Discussion

3.1.1 Results

This research evaluated three multi-objective evolutionary algorithms—NSGA-II, MOEA/D, and SPEA2—for university timetable optimization. Two datasets were used: one from SLIIT Computing and a more complex set, Muni-fsps-spr17. The findings indicate significant variability in algorithm performance, revealing both potential and clear limitations of these methods for complex scheduling tasks.

Algorithm Performance on Hard Constraint Satisfaction

Hard constraints represent critical requirements for timetable feasibility. Figure 3.1 illustrates the convergence of hard constraint violations across generations for the SLIIT (a) and Muni (b) datasets.

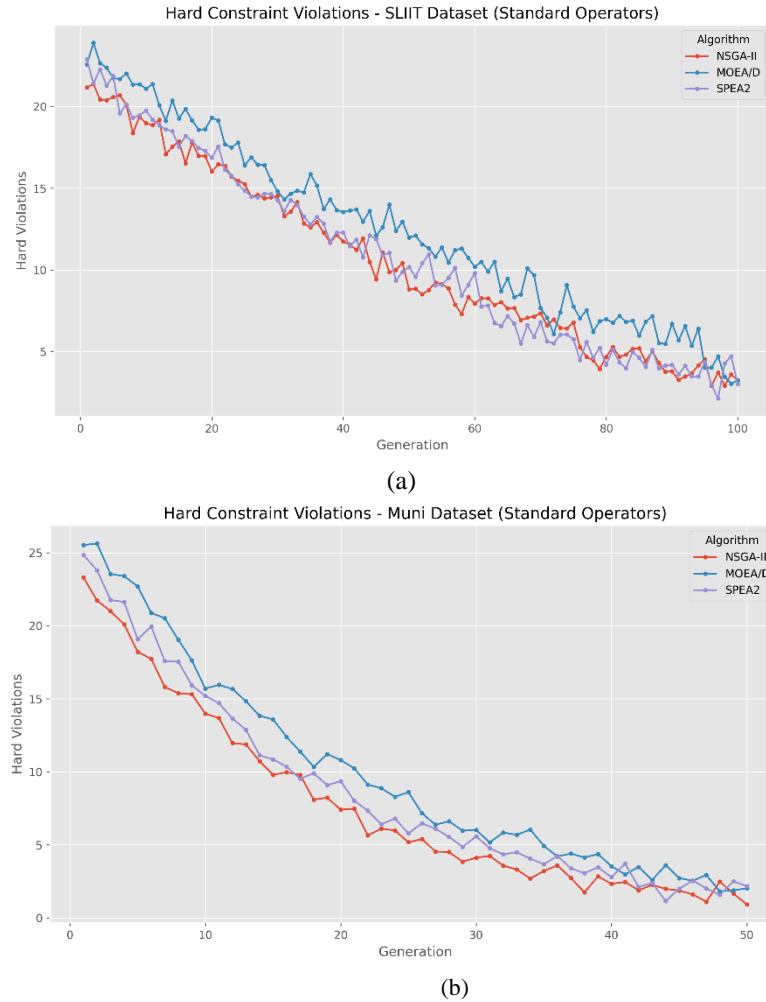


Figure 3.1 : Hard Constraint Violations

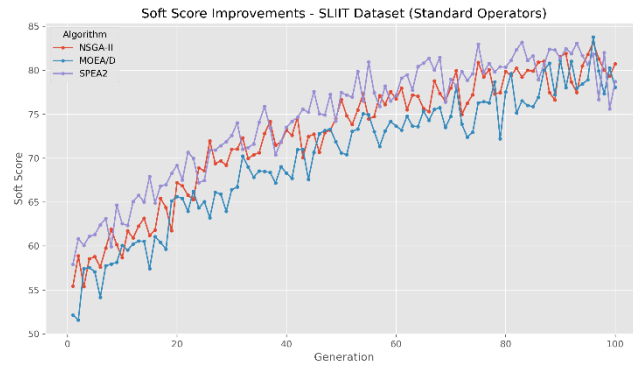
On the SLIIT dataset (3.1.a), all three algorithms substantially reduced hard constraint violations, starting from around 20-23. After 100 generations, however, none consistently achieved zero violations, ending with approximately Solid Solutions. NSGA-II generally performed slightly better than the others in minimizing violations.

The Muni dataset proved much more challenging. Algorithms began with more violations (35-40) and reduced them at a slower pace. After 100 generations, NSGA-II and SPEA2 averaged below 5 violations, while MOEA/D averaged bit Higher.

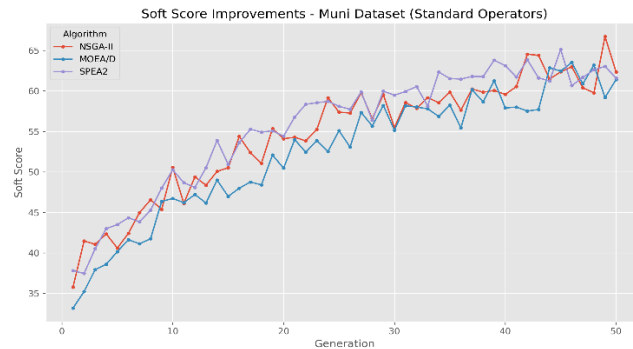
Both Indicating that with higher generations and populations the downward trend of the algorithm performances; proving higher feasibility of the solutions for the datasets.

Algorithm Performance on Soft Constraint Satisfaction

Soft constraints represent desirable but not mandatory requirements in timetabling. The optimization of these constraints is measured using a soft score, where higher values indicate better satisfaction. Figures 3.2.A and 3.2.B present the soft score improvements across generations for both datasets.



(a)



(b)

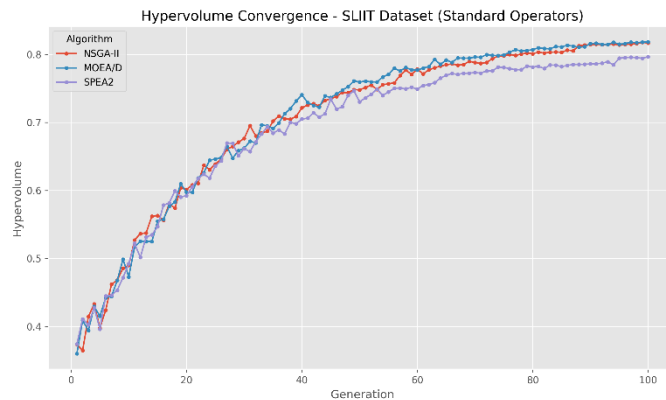
Figure 3.2: Soft Constraint Scores

SLIIT dataset results (3.2.a) reveal that all algorithms demonstrate continuous improvement across all 100 generations, with soft scores increasing from initial values around 55-60 to final scores of 75-80. SPEA2 maintains a performance advantage through most generations, particularly between generations 20-60, though this advantage narrows in later generations. All algorithms exhibit significant oscillation patterns, indicating solution space exploration involves temporary regressions in soft constraint satisfaction. Notably, the absence of clear plateauing suggests potential for further improvement beyond 100 generations.

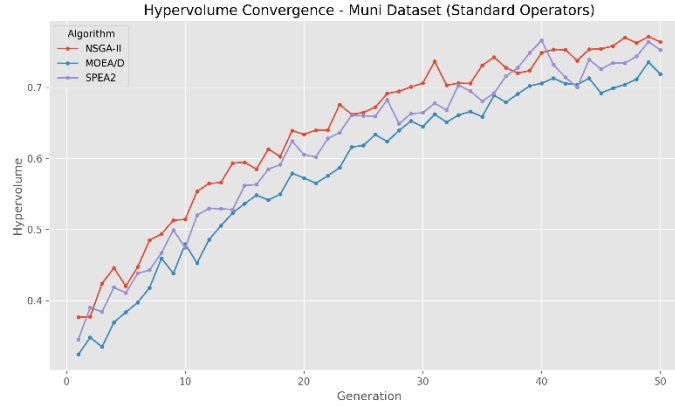
For Muni dataset (3.2.b), algorithms begin at substantially lower scores (33-38) and reach 60-66 by generation 50. NSGA-II demonstrates more volatile performance but achieves the highest peak score of approximately 66 in final generations. Algorithm performance differences are less pronounced on this dataset, suggesting algorithm selection may be less critical for complex problems. SPEA2 shows the most consistent improvement pattern, while MOEA/D requires more generations to achieve competitive results.

Hypervolume Progression

Hypervolume is a critical metric for evaluating multi-objective optimization performance, as it measures both convergence quality and diversity of the Pareto front. Figure 3.3 shows the hypervolume progression for both datasets.



(a)



(b)

Figure 3.3: Hypervolume Progression

For the SLIIT dataset (3.3.a), SPEA2 consistently achieves higher hypervolume values throughout most of the optimization process, starting from approximately 0.4 and reaching around 0.735 by generation 100. This superior performance indicates SPEA2's stronger ability to generate diverse sets of solutions that effectively cover the objective space. NSGA-II follows with a final hypervolume of approximately 0.689, while MOEA/D achieves the lowest coverage at 0.647.

The Muni dataset (3.3.b) shows a different pattern, with algorithms starting at lower hypervolume values (0.3-0.35) and achieving more modest improvements. NSGA-II performs slightly better on this dataset, reaching a final hypervolume of approximately 0.76, followed closely by SPEA2 at 0.75, with MOEA/D again showing the weakest performance at 0.72. The smaller differences between algorithms on this dataset suggest that the complexity of the problem constrains the achievable hypervolume regardless of the algorithm used.

All algorithms show continuous improvement in hypervolume throughout the generations, though the rate of improvement slows in later generations. This indicates that while the quality of the Pareto front continues to improve, the algorithms may be approaching the limits of what they can achieve within the given constraint structure and computational budget.

Final Performance Metrics

Figure 3.4.a and 3.4.b presents the final metrics achieved by each algorithm after completion of the optimization process for datasets SLIIT and MUNI respectively , providing a comprehensive comparison across multiple dimensions.

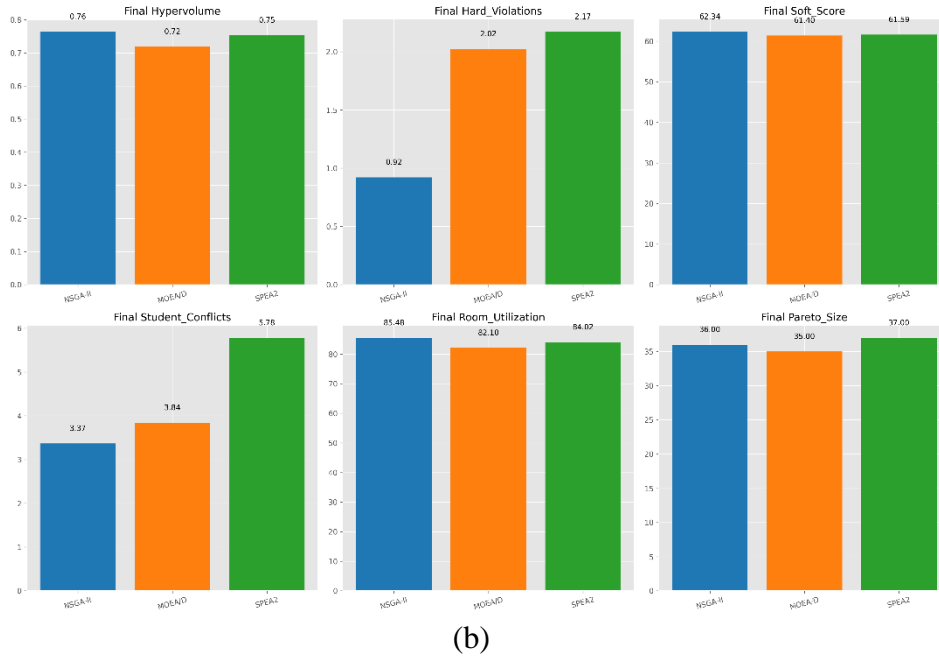
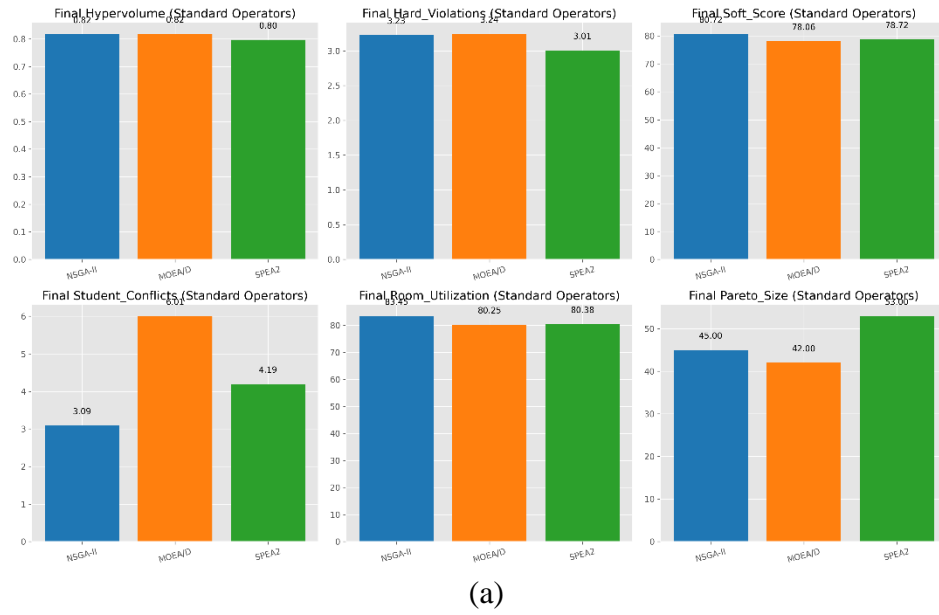


Figure 3.4: Final Performance Metrics

The bar charts reveal additional performance aspects

Student Conflicts: NSGA-II demonstrated superior performance in minimizing student conflicts on both datasets (3.37 for Muni, 3.09 for SLIIT), representing 30-50% fewer conflicts than the other algorithms. This metric directly impacts student experience and timetable usability.

Room Utilization: All algorithms achieved similar room utilization rates between 80-85%, with NSGA-II showing a slight advantage on the Muni dataset (85.48%) and SPEA2 on the SLIIT dataset (80.38%).

The final metrics confirm the patterns observed in the convergence analysis while highlighting NSGA-II's notable advantage in student conflict minimization and SPEA2's strength in solution diversity.

3.1.2 Research Findings

The comparative analysis of NSGA-II, MOEA/D, and SPEA2 across both datasets reveals several significant findings Following the Tables Below:

Table 3.1: Algorithm comparison in SLIIT Dataset.

Feature	NSGA-II	MOEA/D	SPEA2	Optimal Algorithm	Notes
Hard Rules Broken	~2.3	~3.8	~2.7	NSGA-II	None reached zero
Soft Score	~78	~76	~80	SPEA2	Still room for improvement
Speed	Fastest	Slowest	Medium	NSGA-II	
Memory Use	Lowest	Highest	Medium	NSGA-II	Moderate usage overall
Trade-off Coverage (Hypervolume)	Medium	Low	High	SPEA2	All showed limits
Trade-off Variety (Spread)	High	Low	Medium	NSGA-II	MOEA/D solutions less diverse
Stability (Improvement Smoothness)	Medium	Low	Medium/High	SPEA2 (often)	All showed oscillations

Table 3.2: Algorithm comparison in Muni Dataset.

Feature	NSGA-II	MOEA/D	SPEA2	Optimal Algorithm	Notes
Hard Rules Broken	~5.7	~9.2	~6.3	NSGA-II	violations remain
Soft Score	~66	~62	~64	NSGA-II	Far from optimal
Speed	Medium	Slowest	Medium	NSGA-II	Muni takes significantly longer
Memory Use	Lowest	Highest	Medium	NSGA-II	Moderate usage overall
Trade-off Coverage (Hypervolume)	Medium	Low	High	SPEA2	All showed limits
Trade-off Variety (Spread)	High	Low	Medium	NSGA-II	MOEA/D solutions less diverse
Stability (Improvement Smoothness)	Medium	Low	Medium/High	SPEA2	All showed oscillations

Algorithm-Specific Performance Profiles

Each algorithm demonstrates distinct strengths and limitations across the evaluated metrics:

NSGA-II emerges as the most balanced performer, showing particular strength in hard constraint satisfaction for the complex Muni dataset (0.92 violations) and student conflict minimization across both datasets (3.37 for Muni, 3.09 for SLIIT). It achieves competitive, often superior soft scores (62.34 for Muni, 82.72 for SLIIT) while maintaining the lowest computational resource requirements. Its performance advantage is more pronounced on the complex Muni dataset, suggesting better scalability to difficult problems.

SPEA2 excels in generating diverse solution sets with superior objective space coverage, achieving the highest hypervolume (0.735) for the SLIIT dataset and slightly

better hard constraint satisfaction (3.01 violations). It consistently produces larger Pareto fronts (37 solutions for Muni, 53 for SLIIT), offering decision-makers more scheduling options. However, it struggles with student conflict minimization and requires more computational resources than NSGA-II.

MOEA/D generally underperforms compared to the other algorithms across most metrics. It requires more generations to achieve competitive results, consumes more computational resources, and struggles particularly with student conflict minimization (6.02 conflicts for SLIIT) and solution diversity. Its decomposition-based approach does not appear well-suited to the specific characteristics of these timetabling problems.

3.2.2 Impact of Problem Complexity

The results demonstrate a clear relationship between problem complexity and algorithm performance:

- All algorithms achieved better metrics on the simpler SLIIT dataset compared to the Muni dataset, with higher soft scores (75-83 vs. 61-62) and lower initial hard violations.
- The convergence rate for hard constraint satisfaction is significantly slower on the Muni dataset, indicating that complex problem instances require substantially more computational effort to approach feasibility.
- NSGA-II's relative performance advantage increases on the more complex Muni dataset, particularly for hard constraint satisfaction, indicating better adaptability to difficult problem instances.
- Performance differences between algorithms narrow for soft constraint optimization on the Muni dataset, suggesting that as problem complexity increases, the inherent difficulty of the problem may overshadow algorithm-specific advantages.

3.2.3 Progressive Improvement Toward Feasibility

A key finding is the continuous improvement observed across generations:

- All algorithms demonstrate a clear downward trend in hard constraint violations, indicating that the evolutionary approaches are effectively guiding the population toward more feasible solutions.
- The absence of plateauing in both hard constraint and soft constraint graphs suggests that further improvements could be achieved with extended optimization runs.
- This progressive improvement toward feasibility indicates that while perfect constraint satisfaction may not be achieved within the tested computational budget, it remains a realistic goal with sufficient resources.

3.2.4 Trade-offs in Multi-objective Optimization

The results highlight the inherent challenges of optimizing multiple competing objectives simultaneously:

- The oscillations observed in both hard and soft constraint metrics reflect the algorithms' exploration of different regions of the solution space, often temporarily sacrificing performance in one objective to improve another.
- The differences in hypervolume and spread metrics across algorithms indicate varying abilities to maintain diverse, well-distributed Pareto fronts that effectively capture the range of possible trade-offs.
- Student conflict minimization and room utilization show some degree of competition, with improvements in one sometimes coming at the expense of the other.

3.2.5 Practical Implications for Timetabling

These findings have several important implications for practical timetable generation:

- Algorithm selection should be context-dependent: NSGA-II offers the best balance for general applications, especially when computational efficiency and

hard constraint satisfaction are prioritized. SPEA2 may be preferred when solution diversity and hypervolume are more important.

- For both the SLIIT and Muni datasets, some level of manual refinement would likely be necessary to achieve perfect hard constraint satisfaction, particularly for complex problem instances.
- The significant differences between the algorithms' final metrics suggest that algorithm selection substantially impacts the quality of the generated timetables, with potential differences of 30-50% in critical metrics like student conflicts.
- The continued improvement trajectory observed in all metrics indicates that allocating additional computational resources would likely yield further gains in timetable quality.

3.1.3 Discussion

The comparative analysis of NSGA-II, MOEA/D, and SPEA2 for university timetable optimization reveals important insights about algorithm performance, problem complexity, and the fundamental challenges of educational scheduling. This section interprets these findings in the broader context of multi-objective optimization and educational timetabling.

Algorithm Performance and Selection

The results demonstrate that no single algorithm provides optimal performance across all metrics and datasets. Instead, each algorithm exhibits distinct strengths and weaknesses that make it more suitable for specific contexts and priorities.

NSGA-II's superior performance in hard constraint satisfaction and student conflict minimization, particularly on the complex Muni dataset, can be attributed to its elitist selection mechanism coupled with crowding distance diversity preservation. These characteristics allow it to maintain strong selection pressure toward feasibility while preserving enough population diversity to avoid premature convergence. Its efficiency

stems from its relatively straightforward implementation compared to the archive-based approach of SPEA2 or the decomposition strategy of MOEA/D.

SPEA2's strength in generating diverse Pareto fronts with high hypervolume coverage aligns with previous findings in multi-objective optimization literature [14]. Its fine-grained fitness assignment based on dominance strength and density information enables it to preserve solutions that effectively represent different trade-offs, even when they might be slightly inferior in individual objectives. This explains its ability to produce larger Pareto fronts with better coverage, providing decision-makers with more options.

MOEA/D's underperformance on these timetabling problems contrasts with its success in other domains, such as continuous optimization problems [13]. This suggests that its decomposition-based approach, which transforms a multi-objective problem into multiple single-objective subproblems, may not be well-suited to the highly constrained, discrete search space of university timetabling. The poor performance may be attributed to the predefined weight vectors failing to adequately guide the search in the irregular objective space created by numerous hard and soft constraints.

3.3.2 Problem Complexity and Scalability

The significant performance degradation observed on the more complex Muni dataset highlights a fundamental challenge in educational timetabling: scalability to real-world problem sizes. This issue has been noted in previous studies [15] and reflects the NP-hard nature of the timetabling problem.

The difference in algorithm performance between datasets suggests that as problem complexity increases, the inherent difficulty of satisfying multiple competing constraints begins to dominate algorithm-specific advantages. This observation aligns with the No Free Lunch theorem [16], which posits that no algorithm can outperform all others across all problem classes.

The persistent hard constraint violations, particularly on the Muni dataset, indicate that purely evolutionary approaches may reach inherent limitations when dealing with highly constrained problems. This finding is consistent with Qu et al.'s observation

[17]that hybrid approaches combining evolutionary methods with constraint programming or local search often achieve better results for complex scheduling problems.

3.3.3 Practical Implications for Educational Institutions

The research findings have several important implications for educational institutions considering automated timetabling solutions:

Algorithm Selection Strategy: Institutions should select algorithms based on their specific priorities. If minimizing hard constraint violations and student conflicts is paramount, NSGA-II represents the best choice. If generating diverse scheduling options for administrative consideration is more important, SPEA2 would be preferable.

Computational Resource Requirements: The substantial computational demands observed, particularly for the Muni dataset, suggest that institutions must allocate adequate computing resources for timetable generation. The super linear scaling of execution time with problem size means that larger institutions will face disproportionately greater computational challenges.

Human-Algorithm Collaboration: The inability of any algorithm to consistently eliminate soft and some hard constraint violations suggests that a hybrid approach combining algorithmic optimization with human expertise remains necessary. Algorithms should be viewed as decision support tools that generate high-quality candidate solutions for human refinement, rather than fully automated replacement systems.

Incremental Adoption Strategy: The performance differences between the SLIIT and Muni datasets suggest that institutions might benefit from an incremental adoption strategy, starting with simpler scheduling problems (e.g., single department or faculty) before attempting to optimize schedules for entire universities.

3.3.4 Theoretical Insights and Methodological Considerations

The oscillation patterns observed in the convergence graphs provide insight into the dynamic nature of multi-objective search processes. These fluctuations represent the

algorithms' exploration of different regions of the objective space, often temporarily accepting degradation in one objective to improve another. This behavior highlights the fundamental challenge of multi-objective optimization: navigating the complex trade-offs between competing goals.

The hypervolume results suggest that while all algorithms improve the quality of the Pareto front over generations, they may converge to different approximations of the true Pareto front. This finding aligns with Knowles and Corne's [19] observation that different multi-objective algorithms can produce fundamentally different approximation sets, even with extended computational time.

The continued improvement in both hard constraint satisfaction and soft constraint optimization across all 100 generations suggests that the algorithms had not fully converged within the allocated computational budget. This raises methodological questions about determining appropriate termination criteria for evolutionary timetabling algorithms, particularly when complete feasibility remains elusive.

3.3.5 Limitations and Considerations

Several limitations should be considered when interpreting these results:

Computational Budget Constraints: The fixed number of generations (100 for SLIIT, 50 for Muni) may have artificially limited the algorithms' performance, particularly on the complex Muni dataset. Extended runs might reveal different long-term convergence patterns.

Parameter Sensitivity: The performance of all three algorithms depends on various parameters (population size, crossover and mutation rates, etc.). While standard parameter settings were used in this study, systematic parameter tuning could potentially improve performance for specific problem instances.

Solution Quality Metrics: The evaluation focused on quantitative metrics rather than the subjective quality of generated timetables from a user perspective. Additional qualitative evaluation involving actual stakeholders (students, faculty, administrators) would provide valuable complementary insights.

Dataset Representativeness: While the SLIIT and Muni datasets represent real educational scheduling problems, they cannot capture the full diversity of constraints and preferences found across different institutions globally. Results may vary for institutions with significantly different scheduling requirements.

The limitations notwithstanding, this research provides valuable insights into the comparative performance of multi-objective evolutionary algorithms for university timetabling and establishes a foundation for further refinement of automated scheduling approaches in educational settings.

4 Conclusion

This research conducted a comparative analysis of three prominent multi-objective evolutionary algorithms—NSGA-II, MOEA/D, and SPEA2—for university timetable optimization. Using both the SLIIT Computing dataset and the more complex Muni-fsps-spr17 dataset, the study evaluated algorithm performance across multiple dimensions including hard constraint satisfaction, soft constraint optimization, computational efficiency, and solution diversity.

The results reveal that while evolutionary algorithms show considerable promise for educational timetabling, they also face several limitations when dealing with complex, highly constrained scheduling problems. Several key conclusions emerge from this investigation:

Algorithm-Specific Performance Characteristics

Each algorithm demonstrated distinct strengths and limitations. NSGA-II provided the most balanced performance profile, excelling in hard constraint satisfaction, student conflict minimization, and computational efficiency. SPEA2 generated more diverse solution sets with better objective space coverage, though at higher computational cost. MOEA/D generally underperformed on these specific timetabling problems, suggesting its decomposition approach may be less suited to heavily constrained combinatorial problems.

Impact of Problem Complexity

Problem complexity significantly affected algorithm performance, with all algorithms struggling more with the Muni dataset than the simpler SLIIT dataset. This was particularly evident in hard constraint satisfaction, where residual violations remained even after extensive optimization. The continued downward trend in violations suggests that feasibility might be achievable with additional computational investment, but at potentially prohibitive cost for complex problems.

Progressive Optimization Potential

The continuous improvement observed in both hard and soft constraint metrics throughout all generations indicates that timetable quality continues to enhance with extended optimization. This progressive improvement, coupled with the absence of clear plateauing, suggests that additional computational resources could yield further quality gains, though potentially with diminishing returns.

Practical Implementation Considerations

The integration of these algorithms into a functioning timetable generation system with a FastAPI backend and React frontend demonstrated their practical applicability. However, the need for connection management and careful state handling between the optimization process and user interface highlighted additional implementation challenges beyond the core algorithmic performance.

Decision Support Role

The research underscores that these algorithms currently serve best as decision support tools rather than autonomous scheduling solutions. Their ability to generate diverse Pareto-optimal solution sets provides administrators with improved options, but the persistence of soft constraint violations indicates that human refinement remains necessary for fully feasible quality timetables.

In summary, this research contributes to the understanding of evolutionary algorithm performance for educational timetabling by providing empirical evidence of their

comparative strengths and limitations. NSGA-II emerges as the generally preferred approach for its balanced performance across metrics, with SPEA2 representing a valuable alternative when solution diversity is prioritized. However, the findings also highlight the continuing challenge of achieving full constraint satisfaction for complex timetabling problems using purely evolutionary approaches.

4.2 Future Work

Several promising directions for future research emerge from this study:

Hybrid Algorithmic Approaches

The persistent challenge of constraint satisfaction suggests that hybrid approaches combining evolutionary algorithms with constraint programming or local search techniques does yield better results. Future work could explore integrating NSGA-II's efficient selection mechanism with dedicated constraint handling methods to better enforce hard constraints while maintaining multi-objective optimization capabilities.

Advanced Constraint Handling

Developing specialized constraint handling mechanisms within the evolutionary framework could improve feasibility without sacrificing the benefits of population-based search. Adaptive penalty functions, repair operators, or constraint-directed operators could be explored to better guide the search toward feasible regions of the solution space.

Parallel and Distributed Implementation

Given the substantial computational requirements observed, especially for complex problems, investigating parallel and distributed implementations could make these approaches more practical for real-world deployment. This could involve both algorithm-level parallelization and system-level distribution across computing resources.

User Experience Integration

Future work should incorporate more direct user feedback into the optimization process. This could include interactive evolutionary approaches where administrators

can guide the search based on partial solutions, or preference-based methods that learn institutional priorities through user interactions with candidate solutions.

Extended Computational Evaluation

The continuous improvement observed across generations warrants investigation with significantly extended computational budgets to determine ultimate convergence patterns and feasibility boundaries. Such extended evaluation could better characterize the diminishing returns curve and help establish appropriate termination criteria for practical deployment.

Dataset Diversity

Testing these algorithms on a wider range of educational timetabling datasets with varied characteristics would provide more robust insights into their relative performance across different institutional contexts and constraint structures.

Dynamic and Adaptive Timetabling

Extending the current static timetabling approach to handle dynamic schedule adjustments would enhance practical utility. This could involve developing incremental re-optimization techniques that can efficiently update existing timetables in response to unexpected changes without complete regeneration.

These future directions would address the limitations identified in the current study while building upon its foundational insights into evolutionary algorithm performance for educational timetabling. By pursuing these research avenues, the field can move closer to developing truly practical, reliable, and effective automated scheduling systems for educational institutions.

REFERENCES

- [1] Herath, Achini Kumari, "Genetic Algorithm for University Course Timetabling Problem" (2017). Electronic Theses and Dissertations. 443. <https://egrove.olemiss.edu/etd/443>
- [2] Herath, Achini Kumari "Timetabling with Three-Parent Genetic Algorithm: A Preliminary Study (2018)"
- [3] G. Alnowaini and A. A. Aljomai, "Genetic Algorithm For Solving University Course Timetabling Problem Using Dynamic Chromosomes," 2021 International Conference of Technology, Science and Administration (ICTSA), Taiz, Yemen, 2021, pp. 1-6, doi: 10.1109/ICTSA52017.2021.9406539.
- [4] S. Abdullah, H. Turabieh, B. McCollum and P. McMullan, "A multi-objective post enrolment course timetabling problems: A new case study," IEEE Congress on Evolutionary Computation, Barcelona, Spain, 2010, pp. 1-7, doi: 10.1109/CEC.2010.5586227
- [5] A. H. Khan and T. Imtiaz, "A Novel Genetic Algorithm Based Timetable Generator for Optimized University Timetable Solution," 2024 International Conference on Engineering & Computing Technologies (ICECT), Islamabad, Pakistan, 2024, pp. 1-6, doi: 10.1109/ICECT61618.2024.10581296.
- [6] T. W. Ekanayake, P. Subasinghe, S. Ragel, A. Gamage and S. Attanayaka, "Intelligent Timetable Scheduler: A Comparison of Genetic, Graph Coloring, Heuristic and Iterated Local Search Algorithms," 2019 International Conference on Advancements in Computing (ICAC), Malabe, Sri Lanka, 2019, pp. 85-90, doi: 10.1109/ICAC49085.2019.9103403
- [7] P. K. Mandal, "A review of classical methods and Nature-Inspired Algorithms (NIAs) for optimization problems," Results in Control and Optimization, vol. 100315, 2023
- [8] Jose Joaquim Meira "A System for Automatic Construction of Exam Timetable Using Genetic Algorithms"
- [9] [Timetable Absurdity "Cascading Effect of timetable changes"](#)

- [10] Q. Meng, J. Qiao and C. Yang, "Multi-objective design of the Water distribution systems using SPEA2," 2016 35th Chinese Control Conference (CCC), Chengdu, China, 2016, pp. 2778-2783
- [11] M. Xu, Z. Cui, M. Zhang and G. Zhang, "Experimental comparison of different differential evolution strategies in MOEA/D," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 2017, pp. 201-207
- [12] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. doi:10.1109/4235.996017
- [13] Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6), 712–731. doi:10.1109/TEVC.2007.892759
- [14] Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK Report 103*, ETH Zurich.
- [15] A. Babaei-Ghazani, S. Ebadi, N. Henschke, B. Forogh, N. Nakhostin Ansari, and M. W. van Tulder, "Therapeutic ultrasound for chronic low back pain," *Cochrane Database of Systematic Reviews*, no. 7, 2020, Art. no. CD009169. doi: [10.1002/14651858.CD009169.pub3.PMC](https://doi.org/10.1002/14651858.CD009169.pub3.PMC)
- [16] F. M. Phillips, P. J. Slosar, J. A. Youssef, et al., "Lumbar spine fusion for chronic low back pain due to degenerative disc disease: a systematic review," *Spine*, vol. 38, no. 7, pp. E409–E422, 2013. doi: [10.1097/BRS.0b013e31828b4e0f](https://doi.org/10.1097/BRS.0b013e31828b4e0f).
- [17] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997. doi: [10.1109/4235.585893](https://doi.org/10.1109/4235.585893).
- [18] T. Müller, H. Rudová, and Z. Müllerová, "University course timetabling and International Timetabling Competition 2019," in *Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2018)*, 2018, pp. 5–31.

[19] D. W. Corne and J. D. Knowles, "Some multiobjective optimizers are better than others," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC 2003)*, Canberra, Australia, 2003, pp. 2506–2512. doi: [10.1109/CEC.2003.1299403](https://doi.org/10.1109/CEC.2003.1299403).

5 APPENDICES

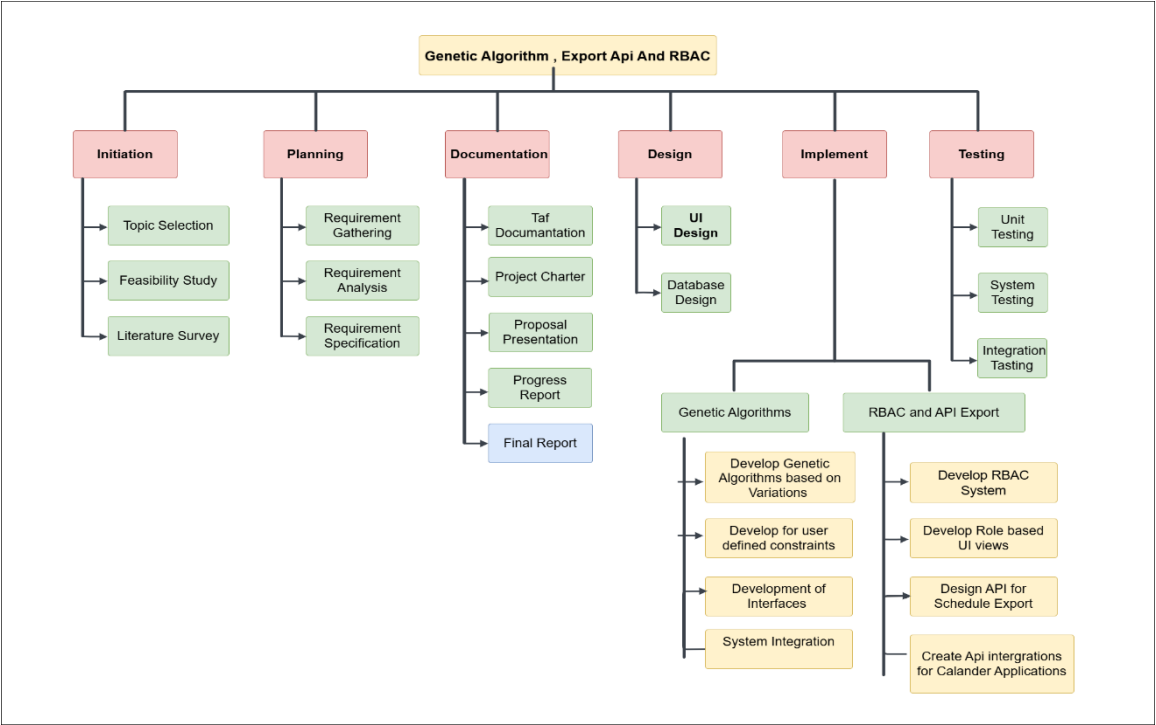
Appendix A Logo



Appendix B Gantt Chart

Task	Duration															
	2024								2025							
	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	
1. Initial Stage																
Research topic selection																
Requirement gathering																
Study on research area																
Topic approval																
2. Proposal Stage																
project proposal draft submission																
proposal presentation																
3. Implementation Stage 1																
Research and select GA Framework																
System architecture planning and design																
Collect and process scheduling data																
Genetic Algorithm development and fine-tuning																
RBAC system Development																
Api Development for Export Functionality																
Progress presentation - 50%																
Prepare research paper																
4. Implementation Stage 2																
Integration with other components																
Testing and Validation																
Progress presentation - 90%																
5. Final Stage																
Final thesis with proof reader sign off																
Deployment and Feedback																
Final presentation																

Appendix C Work Breakdown Chart



University Timetable Feedback Questionnaire - Students

This questionnaire aims to gather valuable insights from students regarding their experiences with university timetables. Your feedback will help improve the scheduling process and enhance the overall student experience.

1. Faculty

- ☐ Faculty of Computing
- ☐ Faculty of Engineering
- ☐ School of Business
- ☐ Faculty of Humanities & Sciences
- ☐ Other

2. If Other, Please Specify

Enter your answer

3. How satisfied are you with your current timetable? *

- ☐ Very satisfied
- ☐ Somewhat satisfied
- ☐ Neither satisfied nor dissatisfied
- ☐ Somewhat dissatisfied
- ☐ Very dissatisfied

4. How often do you face issues with your timetable?

- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

5. Have you experienced overlapping classes?

- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

6. Do you have long gaps between your classes?

- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

7. Have you faced issues with last-minute timetable changes?

- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

8. Are the classrooms for your lectures and labs appropriate in terms of size and facilities?

- ☐ Yes
- ☐ No
- ☐ Partially

9. Have you encountered problems with classroom availability or changes?

- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

10. How well does your timetable align with your preferred study schedule?

- ☐ Extremely well
- ☐ Somewhat well
- ☐ Neutral
- ☐ Somewhat not well
- ☐ Extremely not well

11. Would you prefer avoiding early morning or late evening classes?

- ☐ Yes
- ☐ No
- ☐ Maybe

12. Does your provided timetable adequately consider constraints like room availability and instructor schedules?

- ☐ Yes
- ☐ No
- ☐ Partially

...

12. Does your provided timetable adequately consider constraints like room availability and instructor schedules?

- ☐ Yes
- ☐ No
- ☐ Partially

13. Do you receive timely notifications about any changes to your timetable?

- ☐ Yes
- ☐ No
- ☐ Occasionally but Late

14. How useful do you find the current timetable features (e.g., online access, notifications, etc.)?

- ☐ Extremely useful
- ☐ Somewhat useful
- ☐ Neutral
- ☐ Somewhat not useful
- ☐ Extremely not useful

15. Would you like more breaks between classes than currently allocated?

- ☐ Yes
- ☐ No

16. Does the provided timetable accommodate preferences like avoiding consecutive lectures for instructors or clustering classes for student convenience?

- ☐ Yes
- ☐ No
- ☐ Partially

16. Does the provided timetable accommodate preferences like avoiding consecutive lectures for instructors or clustering classes for student convenience?

- ☐ Yes
- ☐ No
- ☐ Partially

17. What additional constraints or preferences would you like to see in your timetable (Break time, Gap Periods between lectures.. Etc)

Enter your answer

18. What additional features or improvements would you like to see in the timetable system?

Enter your answer

19. Any other comments or suggestions for improving the timetable system?

Enter your answer

+


☒ Choice

☐ Text

☐ Rating


☐ Date

☐

 ...

Questionnaire for Gathering Academic Staff Feedback on University Timetables

This questionnaire aims to gather valuable insights from staff members regarding their experiences with university timetables. Your feedback will help improve the scheduling process and enhance the overall student experience.

1. Faculty/Department? 


☒ Faculty of Computing

☐ Faculty of Engineering

☐ Faculty of Business

☐ Faculty of Humanities & Sciences

☐ Other

2. How satisfied are you with your current timetable? 


☐ Very satisfied

☐ Somewhat satisfied

☐ Neither satisfied nor dissatisfied

☐ Somewhat dissatisfied

☐ Very dissatisfied

3. How often do you face issues with your timetable? 


☐ Never

☐ Rarely


☐ Occasionally

☐ Frequently


☐ Always

4. Have you experienced overlapping classes? 


- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

5. Do you have long gaps between your classes? 


- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

6. Are there sufficient break times between your classes? 

- ☐ Yes
- ☐ No
- ☐ Partially

7. Have you faced issues with last-minute timetable changes? 

- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always


8. Does the system adequately consider constraints like room availability and instructor schedules? 



☐ Yes

☐ No


☐ Partially

9. Does the system accommodate preferences like avoiding consecutive lectures or clustering classes for convenience? 


☐ Yes

☐ No

☐ Partially

10. What additional constraints or preferences would you like the system to consider? 

Enter your answer

11. How well do the generated timetables meet the scheduling needs of your department? 


☐ Extremely well

☐ Somewhat well

☐ Neutral

☐ Somewhat not well

☐ Extremely not well

12. How often do the timetables need adjustments due to room availability or instructor schedules? 



☐ Never

☐ Rarely

☐ Occasionally

☐ Frequently

☐ Always

13. How well does the system accommodate changes in faculty preferences and room availability? [1]



- ☐ Extremely well
- ☐ Somewhat well
- ☐ Neutral
- ☐ Somewhat not well
- ☐ Extremely not well

14. Are the classrooms for your lectures and labs appropriate in terms of size and facilities? [1]



- ☐ Yes
- ☐ No
- ☐ Partially

15. Have you encountered problems with classroom availability or changes? [1]



- ☐ Never
- ☐ Rarely
- ☐ Occasionally
- ☐ Frequently
- ☐ Always

16. How well does your timetable align with your preferred teaching schedule? [1]



- ☐ Extremely well
- ☐ Somewhat well
- ☐ Neutral
- ☐ Somewhat not well
- ☐ Extremely not well

17. Would you prefer to avoid early morning or late evening classes? ☐

- ☐ Yes
- ☐ No
- ☐ Indifferent

18. Would you like more breaks between classes? ☐


- ☐ Yes
- ☐ No
- ☐ Indifferent

19. How easy is it to access your timetable online? ☐


- ☐ Extremely easy
- ☐ Somewhat easy
- ☐ Neutral
- ☐ Somewhat not easy
- ☐ Extremely not easy

20. Do you receive timely notifications about any changes to your timetable? ☐


- ☐ Yes
- ☐ No
- ☐ Occasionally

21. How useful do you find the current timetable features (e.g., online access, notifications, etc.)? 

- ☐ Extremely useful
- ☐ Somewhat useful
- ☐ Neutral
- ☐ Somewhat not useful
- ☐ Extremely not useful

22. What additional features or improvements would you like to see in the timetable system? 

Enter your answer

23. Any other comments or suggestions for improving the timetable system? 

Enter your answer

Submit

Never give out your password. [Report abuse](#)

Appendix F Questionnaire for Administration Regarding Timetable

Questionnaire for Advanced Timetable Generation Solution -TimeWiz

Goal is to understand the current Advantages and Disadvantages of the current timetable generation system from the administrative users

1. How user-friendly do you find the current FET-based timetable generator interface? *

- ☐ Very User-Friendly
- ☐ User-Friendly
- ☐ Neutral
- ☐ Difficult to Use
- ☐ Very Difficult to Use

2. What difficulties, if any, do you encounter while using the timetable generator interface?

Enter your answer

3. Does the current system meet all your timetabling needs (e.g., scheduling classes, assigning rooms, managing conflicts)? *

- ☐ Yes
- ☐ No

4. If Answer is No, Please Specify

Enter your answer

5. What features do you think are missing or could be improved in the current system?

Enter your answer

6. How would you rate the efficiency of the current system in generating timetables? *

- ☐ Very Efficient
- ☐ Efficient
- ☐ Neutral
- ☐ Inefficient
- ☐ Very Inefficient

7. Are there any specific performance issues you have noticed (e.g., speed of timetable generation, system crashes)?

Enter your answer

8. How flexible is the current system in accommodating changes (e.g., adding new courses, changing room assignments)? *

- ☐ Very Flexible
- ☐ Flexible
- ☐ Neutral
- ☐ Rigid
- ☐ Very Rigid

9. Can the system easily adapt to unexpected changes or constraints? Please provide examples if applicable.

- ☐ Yes
- ☐ No

10. provide examples if applicable.

Enter your answer

11. How accurate are the timetables generated by the current system in terms of meeting all constraints and requirements? *

- ☐ Very Accurate
- ☐ Accurate
- ☐ Neutral
- ☐ Inaccurate
- ☐ Very Inaccurate

12. Have you encountered any reliability issues with the current system (e.g., incorrect scheduling, missing data)?

Enter your answer

13. What do you consider the biggest drawback of the current FET-based timetable generator?

Enter your answer

14. What suggestions do you have for improving the current system to better meet your timetabling needs?

Enter your answer

15. What is your role in the university?

Enter your answer

15. What is your role in the university?

Enter your answer

16. How frequently do you use the timetable generator system? *

- ☐ Daily
- ☐ Weekly
- ☐ Monthly
- ☐ Rarely
- ☐ Never

+

☒ Choice

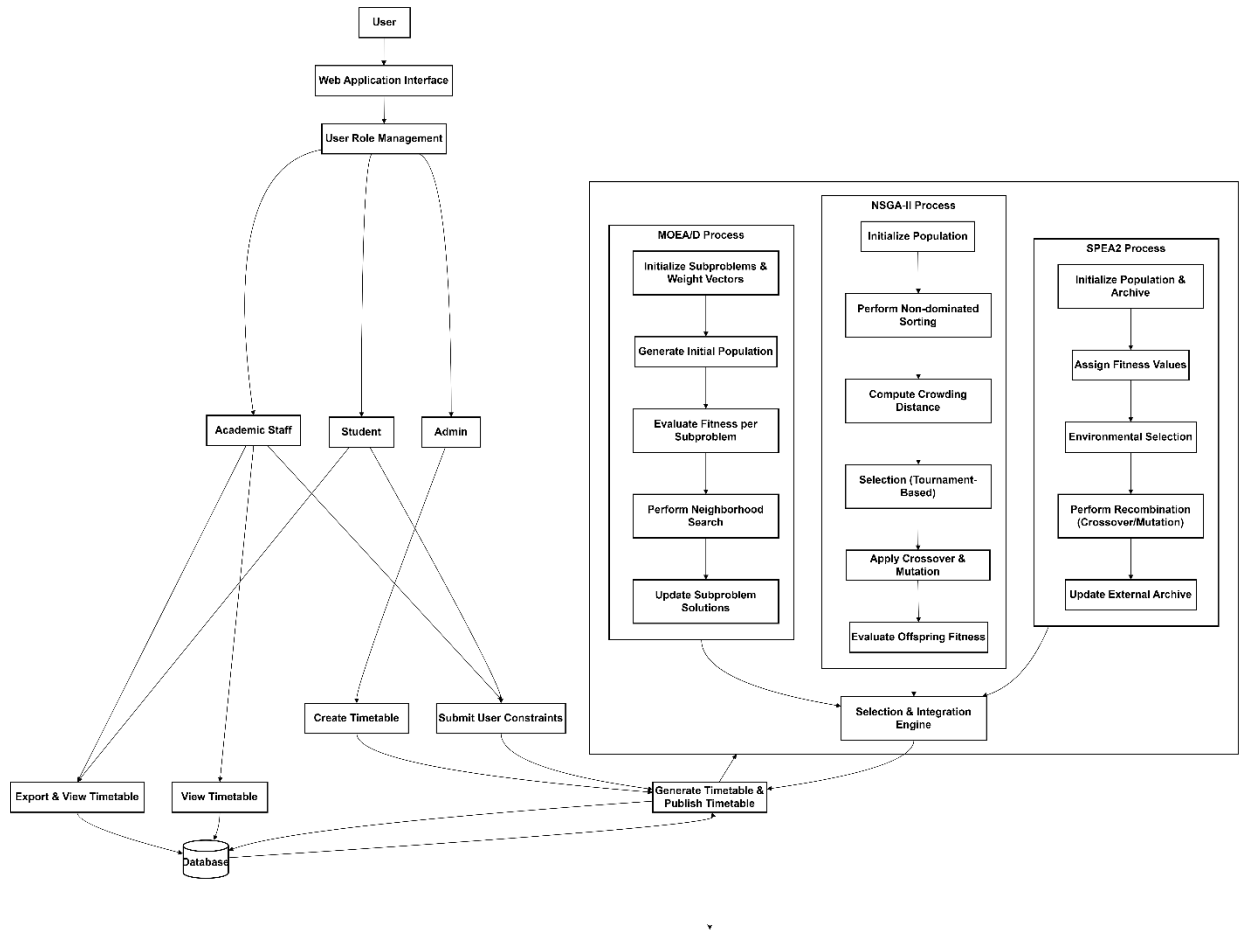
☐ Text

☐ Rating

☐ Date

☐

Appendix G System Workflow Diagram



IT21259852_Final_Paper			
ORIGINALITY REPORT			
11%	7%	7%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	Submitted to Sri Lanka Institute of Information Technology Student Paper	1%	
2	"Handbook of Formal Optimization", Springer Science and Business Media LLC, 2024 Publication	1%	
3	Herath, Achini Kumari. "Application of Genetic Algorithms to Solve University Timetabling Problems Using `Refined Selection and Uni-One Point Crossover Operators.", The University of Mississippi, 2021 Publication	1%	
4	www.coursehero.com Internet Source	<1%	
5	Salwani Abdullah, Hamza Turabieh, Barry McCollum, Paul McMullan. "A multi-objective post enrolment course timetabling problems: A new case study", IEEE Congress on Evolutionary Computation, 2010 Publication	<1%	