

TimeTabler: A Software Framework for Automated Course Timetabling and Space Utilization Optimization in Resource-Constrained Institutions in Bangladesh

Md Abu Jubayer Bijoy

jubayerislam302@gmail.com

Shahjalal University of Science and Technology

Research Article

Keywords: Educational ERP, Genetic Algorithms, ICT4D, Resource-Constrained Environments, Course Timetabling, Software Framework, Space Utilization

Posted Date: October 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11229941/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

TimeTabler: A Software Framework for Automated Course Timetabling and Space Utilization Optimization in Resource-Constrained Institutions in Bangladesh

Md Abu Jubayer Bijoy*

Department of Industrial and Production Engineering,
Shahjalal University of Science and Technology, Sylhet-3114, Bangladesh

Abstract

Educational institutions in developing countries continue to struggle with efficiently scheduling courses due to manual and time-consuming timetabling processes and a lack of optimization tools. While advanced algorithms now exist, software products that can bridge the gap between theoretical optimization and practical administrative use in resource-constrained environments are still lacking. This paper introduces TimeTabler, an open-source desktop application designed to make automated, constraint-aware timetabling accessible to non-technical administrators. The framework is designed for offline, low-cost deployment on low-spec hardware. TimeTabler presents a step-by-step workflow guiding users from data input and constraint configuration to schedule generation and comparison. It incorporates a weighted sum genetic algorithm for optimization. In a deployment case study at a Bangladeshi university involving 10 departments, 56 student batches, and 35 classrooms, the system demonstrated significant potential for operational improvements. The Genetic Algorithm optimizer addressed both quantitative underutilization and qualitative room-type misallocation, raising the overall UFO utilization rate from 52.34% to 59.16% and eliminating all 47 initial room-type mismatches. The software's display of multiple schedules enables administrators to make informed context-sensitive decisions. These findings suggest that optimization-driven scheduling can uncover hidden capacity within existing facilities and support transparent, needs-based resource allocation in resource-constrained institutions.

Keywords: Educational ERP, Genetic Algorithms, ICT4D, Resource-Constrained Environments, Course Timetabling, Software Framework, Space Utilization

1 Introduction

Efficient management of physical resources is a cornerstone of operational efficiency in all stages of educational institutions. Among these resources, teaching and learning spaces such as classrooms, lecture halls, and laboratories are arguably the most critical and require most investment. The efficient utilization of these spaces is directly linked to institutional quality and cost-effectiveness, as well as their ability to accommodate growing student populations [1, 2].

A common challenge faced by institutions around the globe is the perception of classroom shortages. However, this often happens not because there aren't enough spaces but due to poor management and inefficient scheduling [3]. The prevailing manual timetabling method is a tedious and time-consuming process also highly susceptible to human error. Schedulers often find it impossible to simultaneously satisfy a vast number of constraints, ranging from teacher availability and room capacity to student group conflicts, while also maximizing overall space usage and ensuring that specialized facilities are used for their intended purposes. This results

*ORCID: <https://orcid.org/0009-0000-4947-4037>. Email: jubayerislam302@gmail.com

in classrooms being idle for significant periods while other departments struggle to find adequate space or, on the other hand, being severely over-scheduled during peak hours [4].

In a resource-constrained setting like Bangladesh, where public investment in education is limited, optimizing the use of existing infrastructure is a financial necessity [5]. In this case, the issue takes place in two directions. First, there is underutilization in terms of quantity, where spaces are not utilized fully. Second, and most importantly, qualitative misallocation exists, in which the use of spaces is done towards pedagogically unsuitable purposes, such as lecture courses meeting in specialized laboratories or regular classes meeting in auditoriums. This waste of quality resources is a waste of assets and it interrupts the planned learning processes.

Many academic studies have explored advanced algorithms to tackle the university timetabling problem, a known NP-hard optimization issue [6]. However, there is a significant gap between theoretical models and practical tools used by administrators [7, 8]. Metaheuristic methods, especially Genetic Algorithms (GAs), have shown promise in handling complex constraints and improving metrics like room utilization [9, 10, 11]. This issue is particularly serious in resource-limited educational settings. Commercial timetabling systems often involve high licensing fees, require specialized IT support, and function as proprietary “black boxes,” meaning users can’t customize or see how they work [12]. On the other hand, spreadsheet-based methods do not offer optimization features, leading to continued manual inefficiencies. Institutions, especially in developing areas, find themselves stuck between needing accessible solutions and desiring sophisticated ones, unable to apply advanced optimization research practically [13].

To address this gap, this paper introduces TimeTabler. Rather than proposing another algorithm, TimeTabler provides an open-source software framework that makes optimization accessible and usable for non-technical administrators. The main contribution of this work is a deployable desktop framework that integrates a Genetic Algorithm-based optimizer into an accessible, administrator-friendly workflow for data management, constraint configuration, and schedule analysis.

The primary objectives of this study are:

- To design and implement a modular, desktop-based software framework that includes timetabling optimization in a user-friendly workflow for administrators, with offline functionality, low hardware requirements, and free deployment.
- To develop a weighted-sum GA model that accounts for real-world soft and hard constraints, and to apply it to produce timetables that optimize quantitative utilization rates and address qualitative misallocation of room types.
- To evaluate the framework’s practical impact by measuring the current and optimized UFO metrics, operational time savings, and changes in procedures.

By detailing the architecture, design choices, and deployment results of the TimeTabler framework, this work adds a software engineering viewpoint to educational resource management.

This work provides a more holistic and practical interpretation of the management of educational space by proving that it is possible to recover the specialized facilities in the places where they belong and increase the overall efficiency at the same time, through optimization. The conceptual model uses a straightforward input-process-output structure: room and course data serve as inputs, the GA handles optimization, and the outputs are improved utilization metrics and corrected room-type mismatches.

2 Related Work

2.1 Space Management in Educational Institutions

Efficient space management has changed a lot due to digital transformation. It is now the data-driven process of optimizing limited physical resources to support academic activities while keeping costs and environmental impact low [14]. In higher education institutions (HEIs), it includes planning, allocating, and monitoring space to achieve the best possible efficiency. Recent methods emphasize real-time monitoring and flexible allocation in what is called ‘smart campus space management’ [15]. The main goal is to align space use with institutional goals while making sure the facilities are flexible, useful, and fully utilized. Poor space management raises operational costs, disrupts the teaching and learning process, and creates a false shortage of resources [2]. Recent studies show that post-pandemic educational environments need more flexible and efficient space utilization strategies [16].

The UFO model, first introduced by the National Audit Office [17] and improved by recent technology, provides a standard framework for measuring efficiency. The UFO model mainly focuses on two aspects: frequency, which calculates how often a room is used, and occupancy, which measures how full it is when in use. Studies comparing universities in Asia have shown significant differences in utilization rates. This highlights opportunities for improvement [18, 19].

Digital systems such as enterprise resource planning platforms can provide scheduling functions, but they are not always affordable, require infrastructure, and cannot always be optimized [20, 21]. Related challenges have also been observed in advanced learning applications across developing regions [22]. The choice of software design in these situations is firmly related to infrastructure reality. Standalone desktop applications with embedded databases (e.g., SQLite) can operate without a stable network, which is a strong alternative [23, 24, 25, 26]. In addition, usability is of the utmost importance because administrative staff tend to lack technical knowledge, so software should display clear visualizations, provide clear workflow processing, and intuitive configuration interfaces [27]. HCI research demonstrates that decision-support tools are more acceptable when they decrease cognitive load and fit the current system of administration [28, 29].

2.2 Timetabling as an Optimization Problem

Timetabling is a type of scheduling problem that involves assigning resources to objects in space and time while meeting a set of constraints. The university class timetabling problem is NP-hard, meaning it is difficult to find the optimal solution for real-world problems [30, 31]. Recent reviews have categorized timetabling problems into various types based on the context of the institution and the complexity of the constraints [32]. Constraints are usually divided into hard and soft constraints. Hard constraints are strict rules that cannot be broken, such as a teacher being unable to be in two places at once. Soft constraints are desirable qualities that improve the overall practicality of the timetable but are not essential, like avoiding isolated classes for a student group [33]. The goal is to find a workable assignment that meets all hard constraints while minimizing the violation of soft constraints. There is growing attention to multi-objective formulations that balance efficiency and quality metrics [6]. Research in developing countries highlights the need for algorithms that tackle specific local constraints, such as limited technological infrastructure and differing institutional policies [34]. University timetabling research continues to develop, with surveys documenting recent advances in methods and their application areas [35].

Despite this body of research, there remains a significant gap between algorithmic developments and their practical use in educational institutions [7]. Algorithms are typically tested in controlled settings that ignore institutional details [8], and their deployment often fails to

account for the workflow and requirements of administrative employees [36]. As a result, many institutions, especially in developing countries, continue to rely on manual or semi-automated scheduling.

2.3 Genetic Algorithms in Timetabling

Genetic Algorithms are search techniques which are based on the principles of natural evolution and genetics and were first proposed by John Holland in 1975 [37]. They work with a population of possible solutions, known as chromosomes, and each chromosome is assessed by means of a fitness function. Those solutions which are highly fit are chosen to reproduce by means of a process called crossover and also by introducing random changes, a procedure known as mutation. These elements are what are used to produce a new generation and as a result the population tends towards better solutions in each subsequent generation. Nowadays, implementations of GA often include adaptive and hybrid methods in order to make the solutions more robust and to enable real-time changes to the solutions [38].

Genetic algorithms are highly effective for timetabling since they are capable of dealing with complex search spaces that involve multiple constraints [39]. In recent times, machine learning has been incorporated in order to achieve adaptive parameter tuning [40]. Nowadays, many implementations make use of methods which combine genetic algorithms with local search techniques in order to enhance convergence [41, 42, 43]. The application of genetic algorithms in educational institutions has demonstrated their ability to cope with resource constraints without compromising the quality of the solutions [9, 44].

GAs have been successfully used in a variety of timetabling situations, such as the scheduling of university courses and the arrangement of examinations. The research in this area shows that these methods are robust and flexible [30]. The fact that they can be applied in different fields enables them to be adjusted to the particular constraints of various institutions [9]. For this reason, they are suitable for use in the present study, particularly in view of the resource-constrained situation in higher education in Bangladesh.

Although recent studies have concentrated on adaptive parameter tuning through reinforcement learning [40], the present study makes use of a standard GA in order to set up a robust baseline and is concerned with modeling the complex, real-world constraints particular to resource-sharing in developing areas.

2.4 The considerations and trade-offs involved in design

The literature points out that institutions in developing countries still face a significant challenge. They have to choose between accessible but inflexible commercial systems and sophisticated research prototypes which are technically complex. Although commercial solutions have user-friendly interfaces they function as “black boxes”, offering only limited optimization and at high costs [45]. Conversely, the research prototypes, even though they are advanced in terms of their algorithms, seldom consider the infrastructure constraints, the usability needs or the actual workflows of the institutions themselves [8, 36].

The importance of this gap is particularly evident in low-resource situations, since the solutions required there must be simple, cheap, and capable of operating even in the absence of a reliable network infrastructure. They also have to generate results that non-technical administrators can understand and make use of [46, 24]. Furthermore, research into technology adoption shows that people’s perception of how easy a system is to use and the lack of infrastructure are some of the main factors affecting the acceptance of technology in an educational context; systems that are technically competent will not be adopted if usability and suitability to the situation are insufficient [47, 48, 49].

Instead of treating infrastructure independence, a user-friendly workflow, and clear optimization as optional features, TimeTabler focuses on them from the beginning, which stops these

elements from becoming afterthoughts. The framework deliberately strikes a balance between algorithmic power and practical deployment, thereby dealing with the trade-offs that have been mentioned in the literature.

3 Methodology: System Design and Development

3.1 System Architecture Overview

TimeTabler uses a layered architecture to promote modularity, maintainability, and clear separation of concerns. This design allows for the independent development and testing of components.

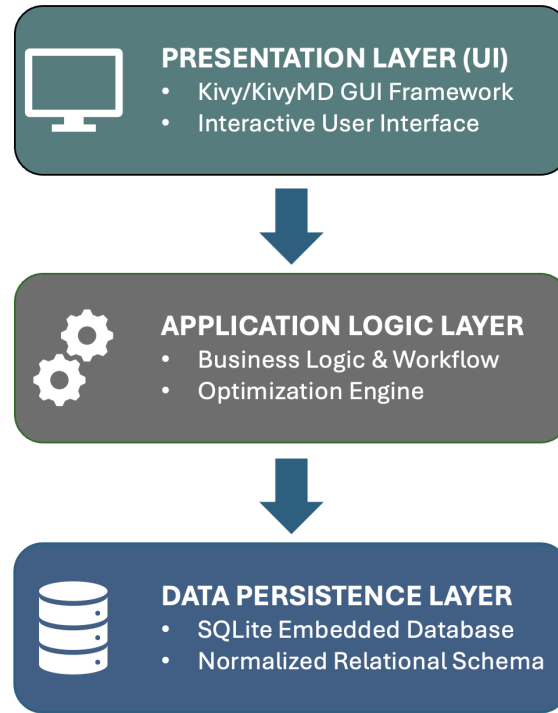


Figure 1: TimeTabler Layered System Architecture

3.2 Technology Stack Selection

The technology stack was selected to align with the core design principles. Table 1 summarizes the key choices and their alignment with design principles.

Table 1: Technology Stack Selection and Rationale

Component	Technology Selected	Key Rationale	Benefit to Users
Core Language	Python 3	Cross-platform, rich libraries, rapid development	Easy maintenance, runs on standard PCs without extra software
GUI Framework	Kivy with KivyMD	Open-source, supports multiple OS	Consistent interface, no licensing costs
Database	SQLite	Lightweight, serverless, zero configuration	Works offline, no IT support needed
Packaging	PyInstaller	Standalone executables, no Python required	Simple deployment on institutional computers

3.3 Data Collection

This study used data from a higher education institution in Bangladesh, gathered through a review of existing class schedules, informal interviews with administrative staff, and direct observation. The dataset covered 10 departments and 35 rooms. The main data points were room capacities, hours of use per week, and student batch sizes for each scheduled class. A standard week was defined as five days from Sunday to Thursday, with eight available hourly slots per day from 8:00 to 17:00, including a lunch break, for a total of 40 available hours per room.

3.4 Performance Measurement: The UFO Model

The Utilization, Frequency, and Occupancy (UFO) model was employed to establish a performance baseline. The calculations are as follows:

$$\text{Frequency Rate (F)} = (\text{Hours Used} / \text{Hours Available}) \times 100 \quad (1)$$

$$\text{Occupancy Rate (O)} = (\text{Total Students} / (\text{Room Capacity} \times \text{Hours Used})) \times 100 \quad (2)$$

$$\text{Utilization Rate (U)} = (F \times O) / 100 \quad (3)$$

This model provides a holistic view, distinguishing between how often a room is used (frequency) and how effectively its capacity is filled when it is used (occupancy).

3.5 Notation and Problem Variables

The timetabling problem was formulated for a resource-sharing environment involving multiple departments with consideration for practical usability.

Table 2: Problem Formulation Variables

Symbol	Definition	Type
$PSet$	Set of all course placements $p \in PSet$	Set
$CSet$	Set of all courses $c \in CSet$	Set
$RSet$	Set of all rooms $r \in RSet$	Set
$TSet$	Set of all teachers $t \in TSet$	Set
$DSet$	Set of all departments $d \in DSet$	Set
$SSet$	Set of all timeslots $s \in SSet$	Set
$LSet$	Set of all Lunchtime slots $l \in LSet$	Set
$teacher(p)$	Teacher assigned to placement p	Function
$room(p)$	Room assigned to placement p	Function
$time(p)$	Timeslot assigned to placement p	Function
$course(p)$	Course associated with placement p	Function
$dept(p)$	Department of course in placement p	Function
$CAP^{-1}(p)$	Number of students in course of placement p	Integer
$REQTYPE(c)$	Required room type for course c	Category
$REQHOURS(c)$	Required hours for course c	Integer
$CAP(r)$	Capacity of room r	Integer
$TYPE(r)$	Type of room r	Category
$L_{actual}(t)$	Assigned hours of teacher t	Integer
$L_{available}(t)$	Available hours of teacher t	Integer
E	Set of timeslots excluded by institutional time policies	Set

3.6 Hard Constraints (H)

H1: Teacher Conflict Constraint. A teacher cannot be assigned to more than one class at the same time.

$$f_1(p, p') = \begin{cases} 1, & \text{if } teacher(p) = teacher(p') \text{ and } time(p) = time(p') \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

H2: Room Conflict Constraint. A room cannot host multiple classes simultaneously.

$$f_2(p, p') = \begin{cases} 1, & \text{if } room(p) = room(p') \text{ and } time(p) = time(p') \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

H3: Room Capacity Constraint. Assigned rooms must accommodate the enrolled student count.

$$f_3(p) = \begin{cases} 1, & \text{if } CAP^{-1}(p) > CAP(room(p)) \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

H4: Room Type Constraint. A course must be assigned to a room that matches its type (e.g., Lab course to a Lab room).

$$f_4(p) = \begin{cases} 1, & \text{if } TYPE(room(p)) \neq REQTYPE(course(p)) \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

H5: Course Hour Requirement Constraint. A course must be scheduled for its exact required number of hours.

$$f_5(c) = \begin{cases} 1, & \text{if } REQHOURS(c) \neq \sum_{p \in PSet, course(p)=c} 1 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

H6: Institutional Time Policies Constraint. A course must not be scheduled in any timeslot excluded by institutional time policies. These excluded timeslots cover three categories:

- Weekends and closed days. Days designated as non-working (e.g., Friday and Saturday) are entirely excluded.
- Daily working hours. Only timeslots within the configured start and end times for each weekday are considered available (e.g., 08:00–17:00). Any timeslot before the start or after the end of the working day is excluded.
- Lunch breaks. Timeslots corresponding to the institution's configured lunch period (e.g., 13:00–14:00) are excluded.

Let E denote the set of excluded timeslots, defined as the union of these three categories. Then:

$$V_6 = \sum_{p \in PSet} f_6(p) = 0, \quad f_6(p) = \begin{cases} 1, & \text{if } time(p) \in E \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

All three categories are user-configurable through the application's settings interface (Table 4).

H7: Faculty Weekly Hours Cap Constraint. An instructor cannot exceed their maximum weekly teaching hours.

$$f_7(t) = \begin{cases} 1, & \text{if } L_{actual}(t) > L_{available}(t) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where $L_{available}(t)$ is the maximum number of weekly teaching hours allocated to instructor t , and $L_{actual}(t)$ is the total number of hours actually assigned to t in the schedule.

H7 enforces a weekly hours cap only. It does not model instructor-specified availability windows or day/time preferences; these are not currently part of the constraint model (see Section 6.5).

H8: Contiguous Timeslot Constraint for Selected Courses. For selected courses such as Lab and Workshop, duration of the course must be met by a single, unbroken block of time.

$$f_8(c) = \begin{cases} 1, & \text{if the placements of course } c \text{ are not in consecutive timeslots} \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

3.7 Soft Constraints (S)

These are desirable and are used to evaluate the quality of feasible solutions.

S1: Course Placement. Maximize the number of courses successfully assigned to rooms and timeslots.

$$\text{Maximize: } SP = \frac{|P|}{|C|} \quad (12)$$

S2: Class Pattern. Classes for the same course should not be scheduled on consecutive days; at least a one-day gap is preferred to aid student learning.

$$\text{Minimize: } C^1 = \sum_{c \in CSet} g^1(c) \quad (13)$$

$$g^1(c) = \begin{cases} 1, & \text{if } \exists d \in \{1, \dots, |D| - 1\} \text{ such that } \text{scheduled}(c, d) \wedge \text{scheduled}(c, d + 1) \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

S3: Faculty Utilization. Maximize the utilization of teacher's available time.

$$\text{Maximize: } C^3 = \frac{1}{|T|} \sum_{t \in TSet} g^3(t) \quad (15)$$

$$g^3(t) = \frac{L_{actual}(t)}{L_{available}(t)} \times 100 \quad (16)$$

3.8 Fitness Function

$$\text{Maximize: } w_1 \times \left(\frac{|P|}{|C|} \times 100 \right) + w_2 \times \left[1 - \left(\frac{C^1}{|P|} \right) \right] \times 100 + w_3 \times C^3 \quad (17)$$

The weight parameters were set to $w_1 = 0.5$, $w_2 = 0.3$, and $w_3 = 0.2$, adding up to 1.0. These values were determined by talking with university administrators to reflect the institution's priorities. Course completion is the most important factor at 50%. This is followed by class pattern quality at 30% and teacher use at 20%.

3.9 Genetic Algorithm Implementation

A Genetic Algorithm optimizer was developed in Python utilizing NumPy and Pandas libraries for efficient data handling and numerical operations.

Chromosome Representation. A direct representation was used. Each gene in a chromosome corresponded to a single class meeting of a course. The gene holds information about the course, assigned room, assigned time slot, and instructor. A chromosome is a collection of these genes, representing a complete weekly schedule. Table 3 represents a chromosome structure for course timetabling problem.

Table 3: Chromosome Representation for Course Timetabling Problem (One Chromosome = One Complete Weekly Timetable)

Gene #	Course ($c \in C$)	Teacher ($t \in T$)	Room ($r \in R$)	Day	Timeslot ($s \in S$)
1	CSE-101	Prof. A	R-201	Sunday	08:00–09:00
2	IPE-301	Prof. B	R-402	Monday	10:00–12:00
3	MATH-202	Prof. C	R-105	Tuesday	11:00–12:00
...	...	...	...	...	...
n	ENG-201	Prof. N	R-407	Thursday	13:00–14:00

Initial Population. The initial population of chromosomes was generated using a limited random generation method. For each course placement, a time slot and a room were randomly picked from the set of feasible combinations that met all hard constraints (H1–H8). If no feasible combination was available for a given placement, the process backtracked by one step to reassign the previous placement. This greedy backtracking method, implemented in Python, created a fully feasible initial chromosome for every individual in the population. The computational overhead of this initialization was negligible, taking less than 2 seconds for a population of 100 chromosomes.

Crossover. A single-point crossover operator was used in this study. Two parent chromosomes are selected. A random crossover point is chosen, and the segments of the chromosomes beyond this point are swapped to create two new offspring.

Mutation. With a low probability (mutation rate), random changes are introduced to the genes of an offspring. This may involve changing the assigned room or time slot for a class. This helps maintain genetic diversity and explore new areas of the search space.

3.10 GA Parameter Selection

The values selected were a population size of 100, a crossover rate of 0.8, and a mutation rate of 0.05, all of which are in accordance with the guidelines for university timetabling problems of comparable complexity [9, 30, 32]. The number of generations was fixed at 500, which is a typical stopping point that provides a good balance between the quality of convergence and computational cost [50]. An elite population percentage of 10% was used, together with tournament selection (using a default tournament size of 3). The process stops when either the maximum number of generations has been reached or a fitness score of 100 (which is the maximum possible fitness) has been obtained. These default settings were applied in the case study that is reported; the software allows for modifications to suit different institutional situations.

In order to take into account the stochastic character of Genetic Algorithms and to assess the consistency of the solutions, the optimization model was run 30 times independently while using the same parameter settings. Summary statistics derived from all of the 30 solutions were

calculated to verify the stability of convergence before the best schedule was chosen for detailed institutional implementation.

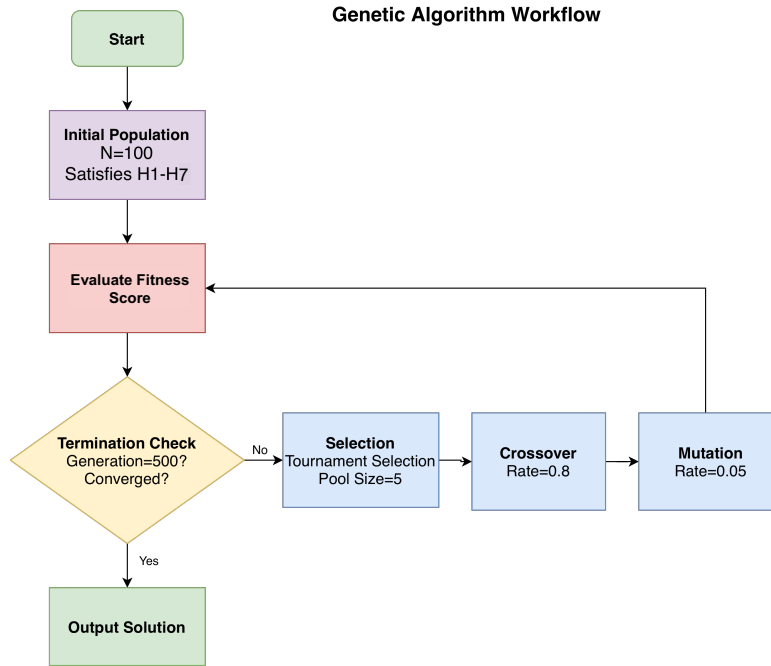


Figure 2: Genetic Algorithm Workflow for Course Timetabling Problem

Figure 2 shows the complete working procedure of genetic algorithm used in this university course timetabling problem.

3.11 User Interface and Administrative Workflow

The user interface features six main steps that follow a logical administrative workflow and it presents optimization through a sequential workflow. Figure 3 illustrates navigation map showing the sequential workflow of TimeTabler (not a formal state machine diagram). Arrows indicate progression from one stage to the next after data validation is completed at each step. Users can return to previous stages via navigation buttons.

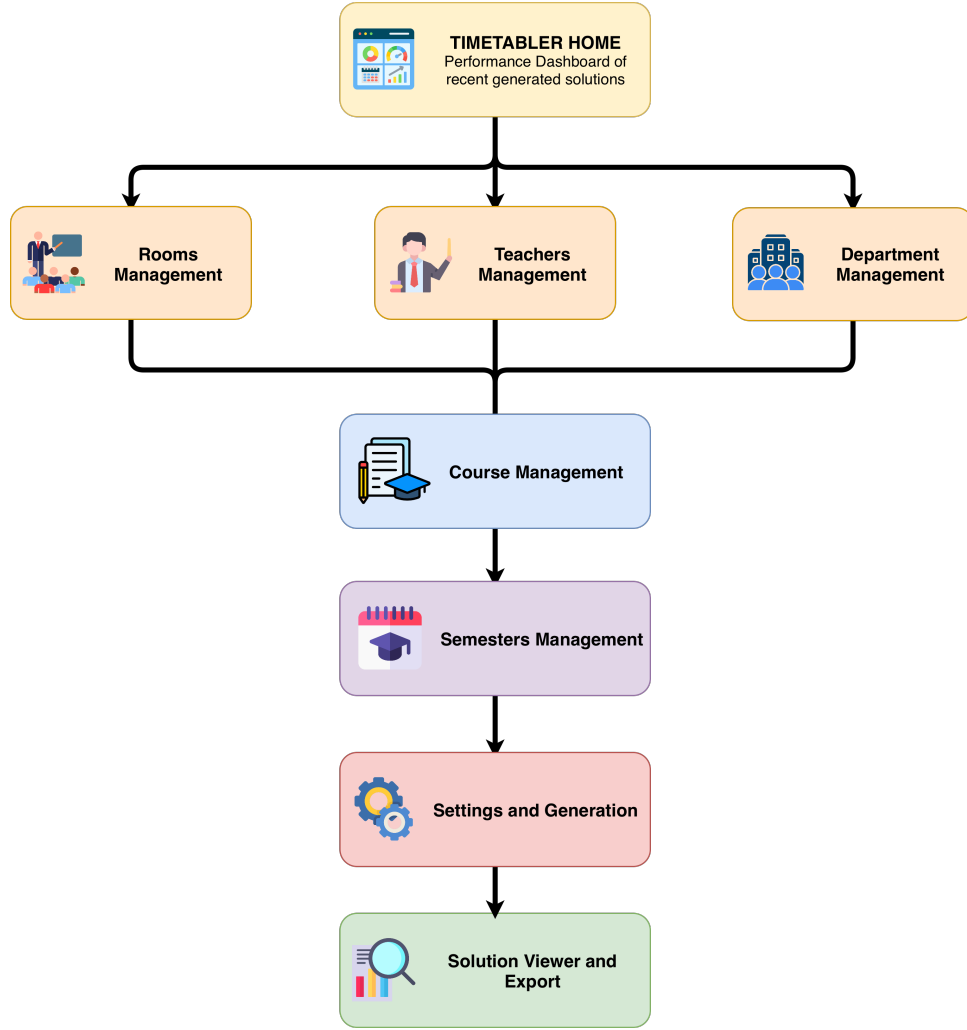


Figure 3: TimeTabler Application Navigation Structure

Workflow Stages:

- **Resource Setup:** Administrators first fill the system with institutional data, including rooms (capacities, types), teachers, and departments.
- **Course Configuration:** Using the room and teacher data already entered, administrators specify for each course its credits, weekly contact hours, students batch size, and any special requirements such as laboratory sessions requiring uninterrupted three-hour blocks or specific room types.
- **Student batch Configuration:** Courses are assigned to specific academic terms, which sets the boundaries for the scheduling problem.
- **Constraint Setup & Generation:** Administrators configure hard and soft constraints based on institutional policies and begin the optimization process.
- **Analysis & Selection:** Generated schedules are compared using visualizations and performance metrics. This helps administrators choose and export the best timetable.

This step-by-step approach reduces cognitive load by presenting one task at a time and ensures all prerequisites are met before moving on to optimization.

4 Implementation

4.1 The approach to development and the architectural framework

The system is based on an event-driven approach in which user actions set off defined workflows via the Kivy framework's event system. Development occurred in stages: first carrying out data modelling and validation, then implementing the constraint management system, followed by the integration of the Genetic Algorithm, and finally the creation of a full user interface. At every stage, feedback was provided by members of the administrative staff in order to enhance the efficiency of the workflow and to make the interface more user-friendly.

4.2 Key Component Implementation Details

4.2.1 Data Management and State Persistence

The system selects SQLite for storing the data locally since no setup is needed in cases where it is not practical to manage a database server. It makes use of SQL Alchemy's ORM for mapping between objects and relational databases, which enables Python classes to stand in for database items such as Room, Instructor, Course, Department, and Semester (Student Batch) with the data being saved automatically. A system for managing state ensures that the data remains consistent throughout the application. This module carries out all the Create, Read, Update, and Delete (CRUD) operations for these entities and guarantees data integrity through real-time validation and by maintaining referential integrity across all entities, thus avoiding orphaned records.

4.2.2 Constraint Modeling and Management Pipeline

The pipeline design of the constraint management subsystem converts the administrator's preferences into optimization parameters.

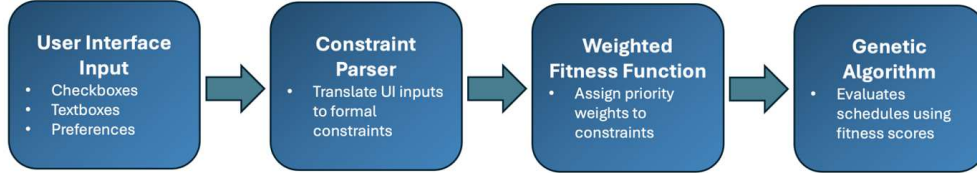


Figure 4: Constraint Processing Pipeline

Figure 4 illustrates how the translation is carried out from UI controls to algorithm parameters. The UI inputs (which include checkboxes and text boxes) are interpreted and converted into formal constraint objects having the form `{'type': 'hard'/'soft', 'rule': 'room_capacity', 'parameters': {...}}`. These objects are then passed to a weighted fitness function, which gives priority weights (for example, course placement is 0.5, class pattern is 0.3, and faculty utilization is 0.2). Finally, the Genetic Algorithm uses these fitness scores to evaluate the schedules.

4.2.3 Optimization Engine Integration

The adapter pattern was used to integrate the Genetic Algorithm optimization engine in such a way that the domain logic remains separate from the algorithmic details, which makes it easier to replace the algorithm in the future. The integration carries out two main types of translation:

that of the domain objects into representations suitable for use in the GA, and that of the GA's output back into actionable schedules together with performance metrics.

Rather than presenting a single “optimal” timetable, TimeTabler generates and displays a number of solutions. Once the optimization process has been finished, the system selects the top five solutions from the last population and shows a number of high-quality alternatives via a comparative interface. With each option are provided the full UFO metrics, so that the various trade-offs can be seen and informed decision-making can be supported.

The design makes particular choices with regard to architecture in order to guarantee good performance as the organization expands. It employs memory-efficient data structures for representing chromosomes and checks the constraints gradually as part of the genetic operations. Because of these features, the system remains responsive even when dealing with large amounts of data. The fact that linear processing times were seen during deployment indicates that the framework will be able to cope with the institution's growth without experiencing major performance problems. Convergence was determined by keeping an eye on the best fitness value over the generations; in reality, the population usually settled down before arriving at the maximum number of 500 generations.

4.3 User Interface Implementation

4.3.1 Interface Implementation Strategy

The interface was developed using Kivy and included KivyMD components, adhering to the Model-View-Controller (MVC) pattern as modified for Kivy's methodology. Each screen is implemented as a separate Kivy Screen class, and when the user carries out an action, this triggers Kivy events (such as `on_press`, `on_text_validate`, and `on_clock_tick`), which are then passed on by the Kivy Event Dispatcher to the appropriate handlers that are responsible for updating the user interface, validating the input, and logging the actions.

4.3.2 Screenshots of the interface and the realization of the design principle

The figures given below display the main screens of the TimeTabler application and show how the design principles were applied to create practical and functional interfaces.

Figure 5: Constraint Configuration Panel

The settings interface illustrated in Figure 5 puts the principles of transparency and user

control into action. The system makes institutional policies—for example, those concerning the avoidance of classes on weekends or scheduling during lunchtime—clear by expressing them through simple textbox entries and checkboxes. This enables administrators to influence the algorithm’s priorities without having to be experts in adjusting the parameters of genetic algorithms. As a result, administrative objectives are effectively linked with the computational optimization.



Figure 6: System Dashboard and Solution Viewer

The dashboard illustrated in Figure 6 presents the comparative metrics (Utilization, Frequency, Occupancy, Fitness Score) for the top five solutions, enabling administrators to compare them and then choose one for export; the second screen displays the schedules on a batch basis and can be navigated using a series of drop-down menus. The design of the interface tackles the problem of resource sharing between departments by making use of visual aids which encourage transparency.

Key implementation decisions included:

- **Responsive Layouts:** Using Kivy’s size hints and adaptive containers to ensure proper

rendering across different screen resolutions.

- **Input Validation:** Immediate feedback for invalid data entry through color coding and error messages.
- **Progress Indication:** Visual progress bars and status updates during long operations.
- **Consistent Navigation:** Maintaining application state during screen transitions.

4.4 Configuration and Adaptability Features

To meet different institutional needs, TimeTabler includes several configuration options. The system has a hierarchical configuration setup where default settings can be overridden by institutional preferences, which can also be overridden by user preferences. This layered approach offers flexibility while keeping sensible defaults.

Table 4: Configuration Layers and Examples

Layer	Example Configuration	Purpose
System Defaults	Default soft constraint weights	Provides working out-of-the-box behavior
Institutional Settings	Working hours: 08:00–17:00, Lunch: 13:00–14:00, Weekends: Friday & Saturday	Captures organization-wide policies
User Preferences	Interface theme: Light, Primary Palette: BlueGray	Customizes user experience
GA Parameters	Population size = 100, Generations = 500, Mutation Rate = 0.05	Customizes optimizer settings

The configuration framework allows for dynamic constraint loading. This feature lets users add new constraint types without having to change the core application code. This flexibility is particularly useful for institutions with unique scheduling needs that standard constraints do not address.

4.5 Performance Optimization and Resource Management

To operate efficiently within common resource limitations, several performance strategies were implemented: lazy loading for data retrieval, indexed database columns for frequent queries, and incremental UI refresh mechanisms. These optimizations enable the application to handle large institutional datasets (dozens of rooms, hundreds of courses) with smooth interaction on standard institutional computers. The framework operates offline with minimal dependencies and maintains compatibility with low-specification hardware.

5 Evaluation and Results

5.1 Deployment Context and Institutional Setting

TimeTabler was deployed at a university in Bangladesh with 10 academic departments sharing 35 classrooms and labs and 56 student batches. The deployment followed one week of software familiarization, two weeks of active use, and systematic assessment of process metrics and schedule outcomes. The software ran on institutional desktops (Intel Core i5-8500, 8GB RAM, Windows 10 Pro) and requires any 64-bit system with ≥ 4 GB RAM.

5.2 Baseline Administrative Process Assessment

Before implementation, timetabling was manual and departmental, using spreadsheets and informal department-to-department negotiations. The process took approximately 12–15 days for a team of several people, consistent with findings reported by Nsulangi et al. [36]. Conflict detection relied on manual cross-checking, and resource allocation was often uneven, with several courses conducted in auditoriums — a hard-constraint violation. This baseline forms the operational reference against which the automated framework is judged.

5.3 Current Space Utilization Performance

For the 10 departments, the overall utilization rate was 52.34% (frequency 62.57%, occupancy 83.65%; Table 5). When rooms were in use, they were usually full, but the frequency rate indicated that rooms sat empty more than a third of the available time. These results match recent studies on space use in universities in developing countries [51].

Table 5: Current UFO’s Rate

Hours Available	Hours Used	Capacity	Total Students	Utilization Rate	Frequency Rate	Occupancy Rate
1400	876	4689	53922	52.34%	62.57%	83.65%

5.4 Optimized Space Utilization via Genetic Algorithm

The weighted-sum GA was executed across 30 independent runs on the institutional dataset involving 56 student batches, 10 departments, and 35 shared rooms. Across all 30 runs, 100% hard constraint satisfaction was achieved. Table 6 presents the aggregate performance metrics across all 30 runs and shows relatively stable performance under the tested configuration.

Table 6: Algorithmic Stability and Performance Across 30 Independent Runs

Performance Metric	Mean $\pm$ SD	Minimum	Maximum
Fitness Score	92.90 $\pm$ 3.51	86.47	97.67
Utilization Rate, U (%)	57.57 $\pm$ 1.22	55.60	59.17
Frequency Rate, F (%)	63.75 $\pm$ 1.36	60.79	65.79
Occupancy Rate, O (%)	90.33 $\pm$ 0.53	89.41	91.56

From this pool of 30 generated solutions, the top-performing candidate schedules were extracted for departmental analysis. Table 7 summarizes top performing chromosomes based on fitness function.

Table 7: Top Performing Chromosomes from Genetic Algorithm Optimization

Chromosome	Fitness	Course Placement	Class Pattern	Faculty Utilization	U (%)	F (%)	O (%)
1	97.67	97.90	96.93	98.20	58.72	65.43	89.75
2	97.22	98.20	96.27	96.17	58.67	65.57	89.48
3	97.38	97.90	96.93	96.75	58.75	65.43	89.79
4	97.38	97.90	95.90	98.29	59.17	65.50	90.33
5	97.09	97.60	97.95	94.53	58.30	65.21	89.41

Table 7 shows that the chromosome with the highest fitness value does not have the highest utilization rate, illustrating the trade-off in the weighted fitness function.

Table 8: UFO Results from Optimum GA Schedule

Hours Available	Hours Used	Capacity	Total Students	Utilization Rate	Frequency Rate	Occupancy Rate
1400	917	4469	40367	59.16%	65.50%	90.33%

Table 8 shows the chromosome with the highest utilization rate. It ranks 4th in fitness value, due to the weighting scheme.

5.5 Statistical Significance Testing

To determine whether the observed improvements are statistically significant, paired t-tests were conducted comparing department-level performance between current and GA-optimized schedules (n=10 departments).

Table 9: Statistical Significance of Improvements

Metric	Current Mean	GA Mean	Mean Difference	SD of Difference	t-statistic	p-value	Cohen's d
Utilization Rate	52.34%	59.16%	+6.82	17.47	0.98	0.351	0.39
Frequency Rate	62.57%	65.50%	+2.93	18.04	0.22	0.827	0.16
Occupancy Rate	83.65%	90.33%	+6.68	7.33	2.85	0.019	0.91

Only Occupancy Rate showed a statistically significant improvement ($p = 0.019$), with a mean increase of 6.68 percentage points and a large effect size ($d = 0.91$) [52]. Utilization and Frequency improved numerically (+6.82 and +2.93 percentage points) but did not reach significance ($p = 0.351$ and 0.827), likely due to the small sample (10 departments) and high variance.

5.6 Comparative Analysis

The comparative analysis showed improvements across all key metrics after implementing the GA-optimized timetable.

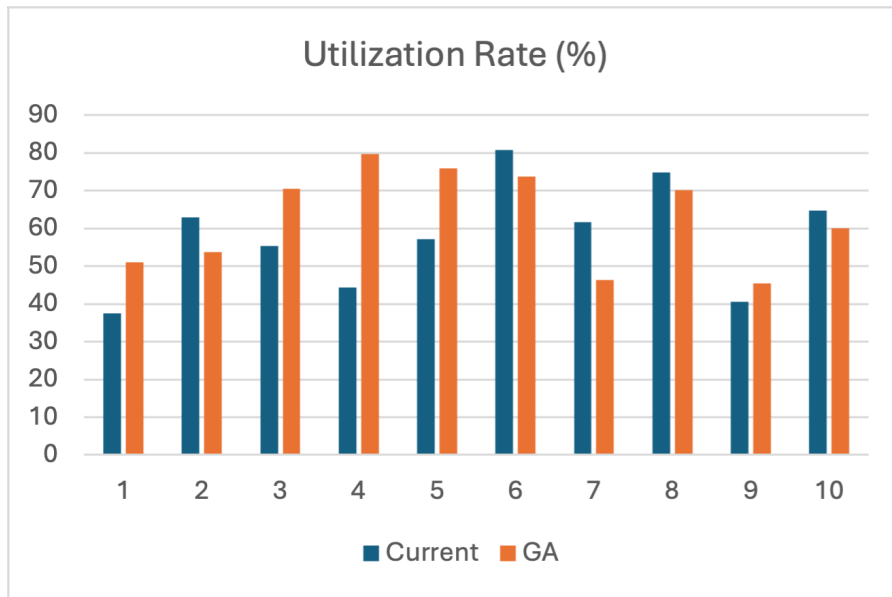


Figure 7: Comparison of Utilization Rate for 10 Departments

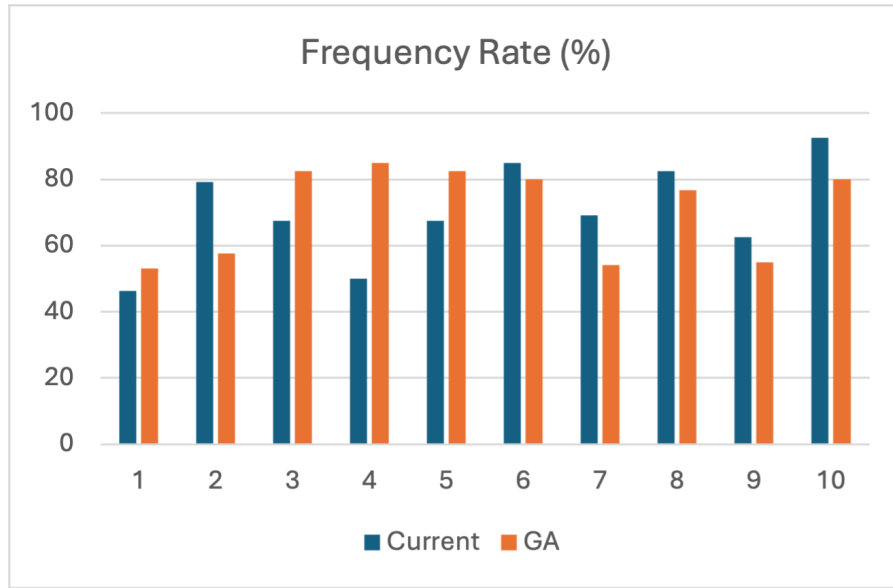


Figure 8: Comparison of Frequency Rate for 10 Departments

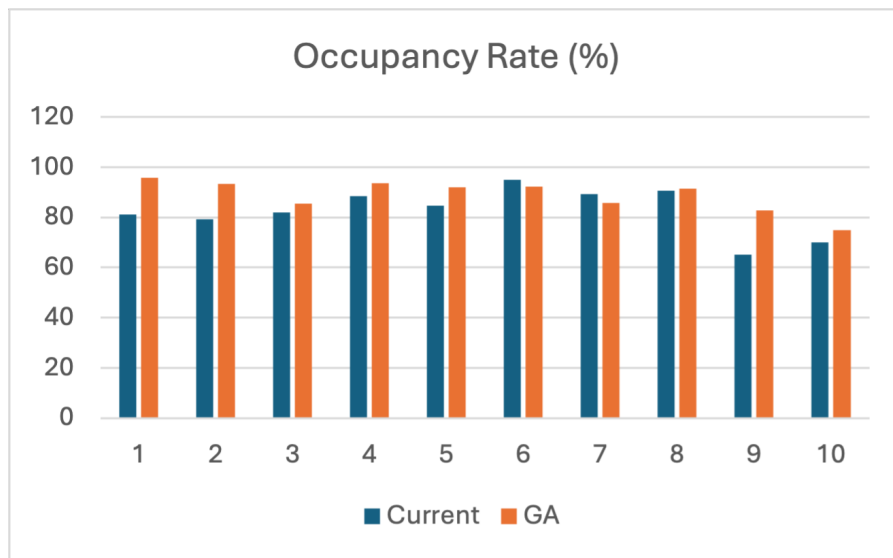


Figure 9: Comparison of Occupancy Rate for 10 Departments

Figures 7–9 show department-wise comparisons for Utilization, Frequency, and Occupancy. Utilization improved from 52.34% to 59.16% (relative +13.03%), Frequency rose slightly by 2.93 percentage points, and Occupancy showed the most dramatic improvement (83.65% to 90.33%).

5.7 Capacity Expansion

Although UFO metrics indicate the efficiency of space usage, they do not directly evaluate what is most important to an institution’s mission—that is, the number of courses it can schedule with its existing resources. The manual scheduling method used 876 weekly room-hours for the 312 scheduled courses, which amounts to 93.69% of the 333 required courses and left 21 course sections without accommodation. The GA-optimized schedule raised the weekly room-hours to 917 and comprised 326 courses, or 97.89%, thus achieving an increase of 14 additional course sections without altering the physical resources.

The effective scheduling capacity is increased by 4.68 per cent because of these 41 extra weekly room-hours; with an average class size of 50, this means 2,050 additional student-contact hours per week, or 34,850 more over the course of a 17-week semester — that is the same as adding 2 to 3 new classrooms without having to make a capital investment.

5.8 Correction of Misallocations of Room Types

One of the key results of this study is that inappropriate room assignments were directly corrected. In the initial schedule, 47 courses, which amounts to 15.1% of the 312 courses that were scheduled, were assigned to rooms that were not suitable for their type; for example, lecture courses took place in computer labs and regular classes were held in auditoriums. However, after applying GA optimization, the number of mismatches was reduced to zero (0 out of the 326 scheduled courses). This full elimination of room-type mismatches was due to the GA's hard constraint H4, which ensured that each course was allocated to the right kind of room. Moreover, once the rooms had been reallocated, there were sufficient standard classrooms to accommodate all the lecture-based courses.

The GA has not only boosted the numerical utilization figures but has also restored specialized facilities, such as laboratories and auditoriums, to their appropriate educational uses. It is now possible for the institution to offer more courses based in laboratories without having to construct new facilities, and students can study in environments which satisfy their educational requirements.

5.9 Operational efficiency and workflow transformation

The use of TimeTabler resulted in substantial savings of time usually spent on scheduling. The average time taken per student batch for automated schedule generation was 15.1 seconds, and this time was almost directly proportional to the size of the problem. That 15.1 seconds figure applies only to the computation time; an extra 3 to 4 hours per semester are needed for data entry across all student batches. Besides being fast, the system centralized scheduling decisions by replacing the previous practice of department-by-department negotiations with clear, system-wide allocation criteria. This increased consistency, reduced the need for manual reconciliation, and made it easier to share common resources transparently. The administrators stated that the time which had previously been used for identifying conflicts and making amendments to spreadsheets could now be redirected towards considering alternative schedule options and assessing institutional priorities.

5.10 User Experience and Deployment Outcomes

The feedback was obtained by means of informal interviews and observations, with no standardized usability tool being employed. Administrators pointed out that the framework's clear and objective criteria for the allocation of resources had dealt with longstanding equity problems which had existed in the previous departmental manual procedures. One administrator in particular noted the greater transparency now available regarding the assignment of rooms and premium facilities, something that stood in sharp contrast to the unclear negotiations that had taken place in the past. The system's ease of use was also shown by the way it integrated smoothly into the existing workflows. It had been mentioned that the time previously spent on settling conflicts could now be put towards evaluating scheduling options and taking account of institutional priorities. This change from a focus on fixing problems to one on forward planning marks a major shift in administrative practice. The findings show that TimeTabler is able to convert optimization algorithms into a tool that non-technical administrators can understand, trust, and use, even in situations where resources are limited. These observations are still preliminary and were gathered through informal interviews without the use of a standardized usability

instrument; they are meant to supplement, not to replace, the formal quantitative evaluation.

6 Discussion

6.1 The Dual-Layer Improvement

The study makes a practical contribution by putting right the qualitative misallocations. In the manual timetable laboratories were being used for regular lectures, this is a hidden form of inefficiency and a waste of specialized equipment. The GA addressed this by introducing the room-type constraint (H4), which released the specialized facilities and allowed the institution to provide more lab based courses without constructing any new facilities, in line with established standards for laboratory facilities [53]. The gradual increase in Frequency Rate is a by-product of this reallocation since courses which were previously held in non-standard, uncounted spaces are now being given in the pool of rooms as measured by the UFO method. Thus, the traditional aim of timetabling optimization is extended beyond simply filling capacity to include functional alignment.

6.2 Framework’s capabilities and design validation

TimeTabler addresses the particular challenges faced by resource constrained institutions. The significant increase in speed changes timetabling from a task that has to be carried out periodically into an interactive planning activity. This ability to generate timetables quickly enables administrators to look at various ‘what-if’ situations, try out different constraint configurations, and swiftly respond to last-minute changes, thus moving the planning approach from one that is reactive to one that is proactive.

The constraint management system is able to deal with a number of conflicting requirements at the same time by taking into account hard constraints, institutional preferences, and optimization objectives, thus arriving at solutions that satisfy the actual needs of the institution rather than those of simplified models.

Presenting the top five solutions acknowledges that scheduling requires value judgments beyond mathematical optimization. It is not the case that every schedule with a high fitness value corresponds to practical preferences or utilization objectives. The five solutions are the best-fitness chromosomes obtained from one run of a genetic algorithm with fixed weights—alternative high-quality schedules, not solutions that have been optimized for different sets of weights.

The design of TimeTabler tackles two problems which affect organizations in developing countries: limited resources and the existence of departmental barriers. Because it has a standalone desktop structure it does not require any servers or continuous connectivity and can therefore be used on standard equipment. The amount of memory it uses at peak load is under 500MB, which means it can operate on older machines and thus prevents the need for expensive upgrades that would hinder the adoption of the technology.

Just as importantly, TimeTabler encourages the sharing of resources among different departments. Since it treats all departments on an equal footing within a single optimization model, it eliminates the “department-first” method which results in suboptimal overall utilization. Inter-departmental conflicts are reduced because clear criteria for allocation together with visual comparison tools make the allocation decisions transparent.

The system is able to handle increasing course loads. Unlike manual methods, in which complexity increases exponentially, or some algorithmic methods that become computationally impractical, TimeTabler keeps its performance at a practical level as the size of the institution grows. The training process takes between two and three hours and all that is required is a basic knowledge of computers.

The fact that the system was deployed proved that the design principles were sound. The system’s ability to function in areas where connectivity is unreliable showed that offline access was feasible. Administrators reported that the system placed less mental strain on them than manual methods, even though a standardized usability scale had not been employed. The fact that administrators could understand and affect the results of optimization showed that both transparency and user control had been achieved. The zero-cost method of deployment was found to be financially viable. Thus, the design principles of the framework have been shown not only to be sound in theory but also to provide effective solutions to actual institutional problems.

6.3 Policy and Practice Implications for Space Management

For university administrators, especially those working in situations where resources are limited, this study provides a fact-driven argument opposing hasty investments in capital. The actual improvements show that a great deal of capacity is being hampered by operational inefficiencies which are of both a quantitative and qualitative nature. Introducing GA-based scheduling turns timetabling into a resource management tool rather than an administrative one. It promotes a fair and need-based distribution of inter-departmental resources and establishes clear allocation criteria that do not rely on historical priority or the power of particular departments. This is particularly important in the case of publicly funded institutions operating in the developing world where the requirement to achieve more with less is a financial necessity.

6.4 Broader implications for educational technology

TimeTabler demonstrates that with careful design it is possible to overcome resource limitations and make advanced tools available in places where they weren’t before. The achievements here extend beyond the field of timetabling; similar approaches could be used to balance the workload of faculty members, allocate research space, and provide advice to students by offering optimization through user-friendly interfaces which take into account infrastructure constraints.

The framework also acts as a model for sustainable educational software development; since it is open-source, modular, and oriented towards the actual needs of institutions rather than being driven by technological fads, it escapes the cycles of obsolescence and dependency which are so common in commercial systems. This aspect is particularly important in developing countries, where long-term sustainability is often more significant than short-term features.

TimeTabler demonstrates that advanced optimization and practical accessibility can exist together. The framework achieves this by combining the power of algorithms with a user-friendly design, taking account of available resources while at the same time maintaining capabilities, and balancing automation with human judgment, thus providing institutions with a straightforward way of improving their administrative processes without having to make much greater use of technical or financial resources.

6.5 Limitations and future work

The study has a number of methodological and technical limitations. It is based on a single anonymized institution in Bangladesh, consisting of 10 departments, 35 rooms, and 56 classes during a 14-week semester, which limits the generalizability of the findings. There was no comparison with existing open-source systems such as FET and UniTime or with benchmark datasets such as ITC-2007 [54]. While the 100% hard constraint satisfaction demonstrates feasibility, it does not indicate comparative optimality. The manual scheduling baseline required 12 to 15 days and user feedback was obtained through informal interviews without using standardized tools such as SUS or measuring time spent. The GA parameters employed were a population size of 100, 500 generations, a mutation rate of 0.05, an elite rate of 10%, and tournament

selection, with no sensitivity analysis carried out. Convergence plots and fitness trajectories are not provided. The weighted-sum GA, with weights of 0.5, 0.3, and 0.2, produces solutions that are optimized for this particular configuration and does not produce a true Pareto front. The constraint model enforces instructor non-overlap and a weekly teaching hours cap, but does not account for instructor-specified availability windows or day/time preferences. The standalone desktop version does not support multi-user collaboration. Scalability beyond the tested configuration (10 departments, 35 rooms) has not been verified; the claim of linear processing time is only valid within this range. The hardware used consisted of an Intel Core i5-8500 with 8GB of RAM and Windows 10 Pro, although the minimum system requirements are 4GB of RAM and a 64-bit operating system. Also, the assumption of 100% student attendance when calculating occupancy is optimistic. Finally, statistical significance was not achieved in the improvements of the Utilization and Frequency rates ($p > 0.05$), probably because of the small sample size of 10 departments and the high variance.

Future research must verify the framework in a number of institutions of varying sizes. In order to set performance benchmarks, direct comparisons should be made with the FET [55], UniTime [56], the ITC-2007 [54] and ITC-2019 [57] datasets. A formal usability evaluation, based on the System Usability Scale and with time-logged baseline data, would offer more solid evidence of efficiency improvements. In order to improve reproducibility and the quality of optimization, the analysis of parameter sensitivity, convergence studies, and the systematic determination of fitness weights should be carried out. The optimization model should be extended to take into account teacher availability windows and preferences, and the weighted-sum GA should be replaced by genuine multi-objective methods such as NSGA-II in order to generate true Pareto frontiers. Later versions might include real-time multi-user collaboration, web portals tailored to specific roles, automated exam scheduling, and IoT-based attendance tracking in order to enable dynamic utilization analytics. It would be useful to carry out a long-term study using a GA-generated timetable in a real-world environment in order to assess its practical usability and to identify any unforeseen problems; furthermore, an automated technique for tuning the parameters could be used to optimize the GA parameters for particular institutional situations. Lastly, research that compares other metaheuristics, for example Simulated Annealing or Ant Colony Optimization, with the same dataset could yield insights into the most efficient and effective algorithm for educational scheduling.

7 Conclusion

The study demonstrates that Genetic Algorithm optimization is able to deal with two kinds of inefficiency in the management of educational space: quantitative underutilization and qualitative misallocation. When a weighted-sum Genetic Algorithm incorporating hard constraints (for example, room-type matching) and soft constraints (such as class patterns and faculty usage) was applied at a university in Bangladesh with limited resources, the improved timetable increased the overall UFO utilization rate by 13.03% (raising it from 52.34% to 59.16%) and eliminated all of the original 47 room-type mismatches. This improvement resulted in an increase in the number of courses offered by 4.48% (from 312 to 326), which is equivalent to adding two or three new classrooms without making any capital investment.

For a long time, the successful introduction of timetabling optimization into higher education has not been limited by the complexity of the algorithms, but rather by the lack of software that enables such capabilities to be used within the structural and human conditions of universities. This article introduces TimeTabler, an open-source, desktop-based system which has been specifically developed to fill this implementation gap.

TimeTabler offers a software engineering approach that demonstrates four key design principles in practice: it includes offline functionality to prevent dependence on an unreliable infrastructure; it features a user-friendly workflow which enables non-technical administrators to

have control; it provides transparent optimization by replacing opaque black-box algorithms with trade-offs that can easily be understood; and it allows for zero-cost deployment so as to remove any financial barrier to adoption. These principles were not merely aspirational but rather formed the basis of the design, and they were put to the test through its implementation at a university in Bangladesh comprising 10 departments, 35 classrooms and 56 student batches.

The results show that there is practical value in this particular situation. TimeTabler cuts down the amount of manual work which had previously taken 12 to 15 days down to just 3 to 4 hours' worth of one-off data entry and about 15.1 seconds of automated generation each student batch, at the same time eliminating all violations of hard constraints and greatly improving compliance with soft constraints. Not only does it increase efficiency, but the new structure also altered the way scheduling decisions were made: rather than holding department-by-department discussions, open and data-based allocation guidelines were introduced which provided transparent allocation criteria that administrators found to be fairer than the earlier manual methods. Apart from saving time, the administrators also noted that the ability to compare several different schedules with clear performance trade-offs enabled more informed decision-making.

For university administrators, particularly those in developing countries, the main point is that they can secure considerable improvements in their operations without costly infrastructure expansion. By using an optimization-driven scheduling approach, hidden capacities in the existing facilities can be identified, room allocation can be ensured in a fair and need-based manner, and specialized rooms such as laboratories and auditoriums can be restored to their original purpose as teaching spaces. Since the GA is capable of dealing with complex, real-world constraints, timetabling becomes a strategic resource management activity rather than merely an administrative one.

The implications of these results extend well beyond the scheduling aspect. TimeTabler demonstrates that advanced computational methods can be successfully applied to software design when software design prioritizes contextual fit over algorithmic sophistication. Because of its modular structure and the fact that it is open source, the framework provides a platform that can be adapted to a wide variety of institutional situations, and the design philosophy itself acts as an example of how to deal with other resource-constrained problems in educational management, for instance those relating to faculty workload management or the allocation of resources in research space.

Several limitations point to further areas for development. The current desktop version does not offer real-time multi-user collaboration, and having it tested in other institutions would increase its generalizability. Upcoming versions could allow direct input of teacher preferences, be extended to cover examination scheduling, and incorporate IoT-based attendance tracking in order to support dynamic utilization analytics.

In the end, TimeTabler does support the main argument that the barrier to optimization in education is not computational but translational. The study provides both a practical tool and a model that can be replicated by demonstrating that user-centered design and algorithmic capability can coexist so as to enable institutions to redirect their limited resources from administrative logistics to their primary purpose, which is teaching and learning. The fact that TimeTabler has proven successful in a real-world situation underscores the value of frugal innovation in educational technology.

Declarations

Ethical Approval: This study was reviewed and approved by the Departmental Academic Committee of the Department of Industrial and Production Engineering at Shahjalal University of Science and Technology. This committee serves as the institutional review body for non-medical research involving human participants within the department. The committee's

authority to review and approve such research is established through the university’s academic governance framework. Ethical clearance was obtained prior to the commencement of data collection. The study involved only anonymized institutional scheduling data and voluntary feedback from administrative staff, with no collection of personal or sensitive information. The committee did not issue a formal waiver reference number or date, as departmental-level reviews at the time did not generate formal documentation. The study was conducted in accordance with the institution’s guidelines for research involving administrative processes.

Consent to Participate: Informed consent was obtained from all individual participants involved in the software deployment case study. Participants were made aware of the study purpose, data usage, and the voluntary nature of their participation.

Consent to Publish: Not Applicable.

Funding: The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing Interests: The authors have no relevant financial or non-financial interests to disclose.

Data Availability: The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Generative AI and AI-Assisted Technologies Declaration: During the preparation of this manuscript, the authors used the following AI-assisted technologies:

- Semantic Scholar and SciSpace were used to assist with literature review and identification of relevant sources. All sources identified were independently verified by the authors for relevance and accuracy.
- DeepSeek-V3 was used for minor grammar correction, language polishing, and formatting suggestions. This included improving sentence clarity and ensuring consistent formatting of references and tables.

All AI-assisted outputs were independently reviewed and verified by the authors. The authors take full responsibility for the accuracy, integrity, and final content of the manuscript. No AI tools were used for data analysis, generation of results, or creation of tables and figures.

References

- [1] A.R. Akinsolu, Provision and management of facilities in Nigerian primary schools, in: J.B. Babalola, E.O. Fagbemiye (Eds.), *Management of Primary and Secondary Education in Nigeria*, NAEAP Publications, Ibadan, 2004.
- [2] H. Shahabudin Abdullah, N.A. Mohd Rahim, Z. Mohd Zain, Space utilisation: Interpreting the result, *Jurnal Teknologi (Social Sciences)* 77 (31) (2015) 55–61. <https://doi.org/10.11113/jt.v77.6868>.
- [3] S.J. Fugazzotto, Mission statements, physical space, and strategy in higher education, *Innovative Higher Education* 34 (5) (2009) 285–298. <https://doi.org/10.1007/s10755-009-9118-z>.

- [4] J. Ye, S. Cai, S. Zhang, N. Wu, Y. Qiao, Space utilization analysis of classrooms for individual studies via back propagation neural network, *Journal of Building Engineering* 102 (2025) 111946. <https://doi.org/10.1016/j.jobe.2025.111946>.
- [5] B. Biswas, S.C. Majumder, Adequacy of resources and quality education: A study at selected public universities of Bangladesh, *Discover Education* 5 (2026) Article 4. <https://doi.org/10.1007/s44217-025-00999-1>.
- [6] A. Bashab, A.O. Ibrahim, I.A. Tarigo Hashem, K. Aggarwal, F. Mukhlif, F.A. Ghaleb, A. Abdelmaboud, Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues, *Computers, Materials & Continua* 74 (3) (2023) 6461–6484. <https://doi.org/10.32604/cmc.2023.034051>.
- [7] E.K. Burke, J. Mareček, A.J. Parkes, H. Rudová, Decomposition, reformulation, and diving in university course timetabling, *Computers & Operations Research* 37 (3) (2010) 582–597. <https://doi.org/10.1016/j.cor.2009.02.023>.
- [8] R.A. Oude Vrielink, E.A. Jansen, E.W. Hans, Practices in timetabling in higher education institutions: A systematic review, *Annals of Operations Research* 275 (1–2) (2019) 145–160. <https://doi.org/10.1007/s10479-017-2688-8>.
- [9] A.R. Mahlous, H. Mahlous, Student timetabling genetic algorithm accounting for student preferences, *PeerJ Computer Science* 9 (2023) e1200. <https://doi.org/10.7717/peerj-cs.1200>.
- [10] K. Xiang, X. Hu, M. Yu, X. Wang, Exact and heuristic methods for a university course scheduling problem, *Expert Systems with Applications* 248 (2024) 123383. <https://doi.org/10.1016/j.eswa.2024.123383>.
- [11] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, *OR Spectrum* 30 (2008) 167–190. <https://doi.org/10.1007/s00291-007-0097-0>.
- [12] M. Sabiteka, X. Yu, C. Sun, Toward sustainable education: A contextualized model for educational technology adoption for developing countries, *Sustainability* 17 (8) (2025) 3592. <https://doi.org/10.3390/su17083592>.
- [13] F.P. Diallo, A.S. Haïdara, M.B. Keita, Optimizing the scheduling of teaching activities in a faculty: Addressing limitations of manual methods and improving resource utilization, *Applied Sciences* 14 (20) (2024) 9554. <https://doi.org/10.3390/app14209554>.
- [14] S. Azizi, G. Nair, R. Rabiee, T. Olofsson, Application of Internet of Things in academic buildings for space use efficiency using occupancy and booking data, *Building and Environment* 186 (2020) 107355. <https://doi.org/10.1016/j.buildenv.2020.107355>.
- [15] B. Valks, M.H. Arkesteijn, A.C. Den Heijer, H.J.M. Vande Putte, Smart campus tools: Adding value to the university campus by measuring space use real-time, *Journal of Corporate Real Estate* 20 (2) (2018) 103–116. <https://doi.org/10.1108/JCRE-03-2017-0006>.
- [16] L. Barua, B. Lockee, A review of strategies to incorporate flexibility in higher education course designs, *Discover Education* 3 (2024) 127. <https://doi.org/10.1007/s44217-024-00213-8>.
- [17] National Audit Office (NAO), *Space Management in Higher Education: A Good Practice Guide*, HMSO, London, 1996.
- [18] A.A. Hasmadi, N. Kamaruddin, N.A.A. Agus Salim, Space utilisation rate of public higher education classrooms in Klang Valley, *Built Environment Journal* 23 (1) (2026) 22–31. <https://doi.org/10.24191/bej.v23i1.1106>.

- [19] S. Osei, J. Quansah, A. Abudu, Management of teaching space and time: Evidence from tertiary institutions in Ghana, *Advances in Social Sciences Research Journal* 11 (8) (2024) 240–264. <https://doi.org/10.14738/assrj.118.17426>.
- [20] M.B. Sholeh, R.F. Samodra, A.P. Widodo, Benefits and challenges of ERP implementation in higher education institutions: A systematic literature review, *Jurnal Sistem Informasi Bisnis* 15 (1) (2025) 21–33. <https://doi.org/10.14710/vol15iss1pp21-33>.
- [21] K. Sigama, B.M. Kalema, Model of cloud ERP adoption in developing countries' institutions of higher learning, in: *Impacts of Technology on Operations Management*, IGI Global, 2025, pp. 149–178. <https://doi.org/10.4018/979-8-3693-6205-1.ch006>.
- [22] N. Eltaiba, S. Hosseini, K. Okoye, Benefits and impact of technology enhanced learning applications in higher education in Middle East and North Africa: A systematic review, *Global Transitions* 7 (2025) 350–374. <https://doi.org/10.1016/j.glt.2025.06.004>.
- [23] R. Heeks, Information systems and developing countries: Failure, success, and local improvisations, *The Information Society* 18 (2) (2002) 101–112. <https://doi.org/10.1080/01972240290075039>.
- [24] A. Agbeyangi, H. Suleman, Advances and challenges in low resource environment software systems: A survey, *Informatics* 11 (4) (2024) 90. <https://doi.org/10.3390/informatics11040090>.
- [25] S. Barikzai, V. Bharathi, A. Perdana, Challenges and strategies in e-learning adoption in emerging economies: A scoping review, *Cogent Education* 11 (1) (2024) 2400415. <https://doi.org/10.1080/2331186X.2024.2400415>.
- [26] G. Walsham, ICT4D research: Reflections on history and future agenda, *Information Technology for Development* 23 (1) (2017) 18–41. <https://doi.org/10.1080/02681102.2016.1246406>.
- [27] R. Kruger, J. Brosens, M. Hattingh, A methodology to compare the usability of information systems, in: *Responsible Design, Implementation and Use of Information and Communication Technology*, Vol. 12067, Springer, 2020, pp. 452–463. https://doi.org/10.1007/978-3-030-45002-1_39.
- [28] A. Adem, E. Çakıt, M. Dağdeviren, Selection of suitable distance education platforms based on human–computer interaction criteria under fuzzy environment, *Neural Computing and Applications* 34 (10) (2022) 7919–7931. <https://doi.org/10.1007/s00521-022-06935-w>.
- [29] A. Mirabdollah, M. Alaeifard, A. Marandi, User-centered design in HCI: Enhancing usability and interaction in complex systems, *International Journal of Advanced Human Computer Interaction* 2 (1) (2023) 16–33. <https://www.ijahci.com/index.php/ijahci/article/view/16>.
- [30] H. Babaei, J. Karimpour, A. Hadidi, A survey of approaches to the university course timetabling problem, *Computers & Industrial Engineering* 86 (2015) 43–59. <https://doi.org/10.1016/j.cie.2014.11.010>.
- [31] X. Gu, M. Krish, S. Sohail, S. Thakur, F. Sabrina, Z. Fan, From integer programming to machine learning: A technical review on solving university timetabling problems, *Computation* 13 (1) (2025) 10. <https://doi.org/10.3390/computation13010010>.
- [32] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, V.B.S. Prasath, Choosing mutation and crossover ratios for genetic algorithms—A review with a new dynamic approach, *Information* 10 (12) (2019) 390. <https://doi.org/10.3390/info10120390>.

- [33] N. Pillay, A survey of school timetabling research, *Annals of Operations Research* 218 (2014) 261–293. <https://doi.org/10.1007/s10479-013-1321-8>.
- [34] H. Chikwiriro, C. Pharaoh, M. Prudence, D. Ndumiyana, Use of genetic algorithms in rigorous university timetabling, *International Journal of Engineering Research & Technology (IJERT)* 3 (3) (2014).
- [35] S. Ceschia, L. Di Gaspero, A. Schaerf, Educational timetabling: Problems, benchmarks, and state-of-the-art results, *European Journal of Operational Research* 308 (1) (2023) 1–18. <https://doi.org/10.1016/j.ejor.2022.07.011>.
- [36] P.T. Nsulangi, W.E. Ngongi, M.R. Likamba, O.B. Sarehe, M.A. Mkwande, A comparative analysis of manual and automatic timetabling approaches for resource utilisation in tertiary higher learning institution, *International Journal of Computer Science and Mobile Computing* 13 (12) (2024) 65–76. <https://doi.org/10.47760/ijcsmc.2024.v13i12.007>.
- [37] J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, MI, 1975.
- [38] T.A. El-Mihoub, A.A. Hopgood, L. Nolle, Self-adaptive learning for hybrid genetic algorithms, *Evolutionary Intelligence* 14 (2021) 1565–1579. <https://doi.org/10.1007/s12065-020-00425-5>.
- [39] W.J. Wang, Improved adaptive genetic algorithm for course scheduling in colleges and universities, *International Journal of Emerging Technologies in Learning (iJET)* 13 (6) (2018) 29–42. <https://doi.org/10.3991/ijet.v13i06.8442>.
- [40] H. Xu, H. Cheng, M. Tong, Study on genetic algorithm of course scheduling based on deep reinforcement learning, *Computer Science* 51 (6A) (2024) 230600062–8. <https://doi.org/10.11896/jsjx.230600062>.
- [41] Q. Zhang, An optimized solution to the course scheduling problem in universities under an improved genetic algorithm, *Journal of Intelligent Systems* 31 (1) (2022) 1065–1073. <https://doi.org/10.1515/jisys-2022-0114>.
- [42] V. Králev, Combining genetic algorithm with local search method in solving optimization problems, *Electronics* 13 (20) (2024) 4126. <https://doi.org/10.3390/electronics13204126>.
- [43] N. Pillay, A review of hyper-heuristics for educational timetabling, *Annals of Operations Research* 239 (1) (2016) 3–38. <https://doi.org/10.1007/s10479-014-1688-1>.
- [44] E.A. Abdelhalim, G.A. El Khayat, A utilization-based genetic algorithm for solving the university timetabling problem (UGA), *Alexandria Engineering Journal* 55 (2) (2016) 1395–1409. <https://doi.org/10.1016/j.aej.2016.02.017>.
- [45] J.S. Alqurni, Evaluating the user interface and usability approaches for e-learning systems, *International Journal of Information Technology and Web Engineering (IJITWE)* 18 (1) (2023) 1–25. <https://doi.org/10.4018/IJITWE.333638>.
- [46] V.L. Sithole, I. Mbukanma, Prospects and challenges to ICT adoption in teaching and learning at rural South African universities: A systematic review, *Research in Social Sciences and Technology* 9 (3) (2024) 178–193. <https://doi.org/10.46303/ressat.2024.54>.
- [47] A. Granić, Educational technology adoption: A systematic review, *Education and Information Technologies* 27 (2022) 9725–9744. <https://doi.org/10.1007/s10639-022-10951-7>.

- [48] M. Mukred, U.A. Mokhtar, B. Hawash, H. AlSalman, M. Zohaib, The adoption and use of learning analytics tools to improve decision making in higher learning institutions: An extension of the technology acceptance model, *Heliyon* 10 (4) (2024). <https://doi.org/10.1016/j.heliyon.2024.e26315>.
- [49] T. Teo, Factors influencing teachers' intention to use technology: Model development and test, *Computers & Education* 57 (4) (2011) 2432–2440. <https://doi.org/10.1016/j.compedu.2011.06.008>.
- [50] K.L. Mills, J.J. Filliben, A.L. Haines, Determining relative importance and best settings for genetic algorithm control parameters, *Evolutionary Computation* 23 (2) (2015) 309–342. https://doi.org/10.1162/EVCO_a_00137.
- [51] S. Osei, J.Y.D. Quansah, M.A. Mensah, Assessing and improving teaching space utilization for operational efficiency: A study in a Ghanaian tertiary institution, *European Journal of Contemporary Education and E-Learning* 3 (4) (2025) 79–102. [https://doi.org/10.59324/ejceel.2025.3\(4\).06](https://doi.org/10.59324/ejceel.2025.3(4).06).
- [52] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed., Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.
- [53] National Research Council, 6 Facilities, equipment, and safety, in: *America's Lab Report: Investigations in High School Science*, The National Academies Press, Washington, DC, 2006, pp. 168–196. <https://doi.org/10.17226/11311>.
- [54] L. Di Gaspero, B. McCollum, A. Schaerf, The Second International Timetabling Competition (ITC-2007): Curriculum-based course timetabling (Track 3), in: F. Pecora, N. Policella (Eds.), *Proceedings of the 1st International Workshop on Scheduling a Scheduling Competition (SSC 2007)*, 2007. <http://pst.istc.cnr.it/RCRA07/articoli/P08-digaspero-et-al-RCRA07.pdf>.
- [55] L. Lalescu, V. Dirr, FET: Free Timetabling Software (Version 7.10.5) [Computer software], 2026. <https://lalescu.ro/liviu/fet/>.
- [56] T. Müller, UniTime: University Timetabling System [Computer software], 2024. <https://www.unitime.org/>.
- [57] T. Müller, H. Rudová, Z. Müllerová, Real-world university course timetabling at the International Timetabling Competition 2019, *Journal of Scheduling* 28 (2025) 247–267. <https://doi.org/10.1007/s10951-023-00801-w>.