وقائع مؤتمرات جامعة سبها
Sebha University Conference Proceedings
Confrence Proceeeding homepage: http://www.sebhau.edu.ly/journal/CAS

A Genetic Algorithm Approach for University Course Timetabling: A Case Study at the Faculty of IT, Sebha University

*Aisha Mousa Al-Fitouri[a], Asma Agaal[a], Moataz Mohamed Ali[b], Hamadi Ahmed Bugrin[c]

[a, c, d] Computer Science Department, Faculty of Information Technology, Sebha University, Sebha, Libya
[a]Artificial Intelligence Department, , Faculty of Technical Sciences, Sebha, Libya

Keywords:
University Timetabling
Genetic Algorithm
Scheduling
Optimization
Constraint Satisfaction
Educational Technology
Sebha University
Libya

A B S T R A C T

University course timetabling is a complex optimization problem, crucial for the efficient operation of educational institutions. Manual methods are often time-consuming, error-prone, and struggle to handle the increasing complexity of constraints and resources, leading to conflicts and inefficiencies. This paper presents the development and implementation of an automated timetabling system for the Faculty of Information Technology at Sebha University, Libya, utilizing a Genetic Algorithm (GA). The system aims to generate feasible and optimized class schedules by considering various hard and soft constraints, including lecturer availability, room capacity, course requirements, and avoiding clashes for students and lecturers within the same timeslot. The system was developed using Python with the Django framework and SQLite database. Experiments were conducted using real data from the faculty. The results demonstrate that the GA-based system successfully generated conflict-free timetables (zero hard constraint violations) within a reasonable computation time (approx. 193-257 seconds) and number of generations (approx. 101-126). This automated approach provides a significant improvement over the faculty's previous manual process, which often required several days of work and still resulted in persistent scheduling conflicts**.** This work significantly improves upon manual methods and addresses the specific challenges faced by the faculty.

نهج الخوارزمية الجينية لجدولة المقررات الجامعية: دراسة حالة في كلية تكنولوجيا المعلومات، جامعة سبها

عائشة موسى الفيتوري[a*]، أسمه اعجال[a]، معتز محمد علي[b]، حمادي أحمد بوقرين[c]

[a,b,c] قسم علوم الحاسوب، كلية تكنولوجيا المعلومات، جامعة سبها، ليبيا
[a] قسم الذكاء الاصطناعي، كلية العلوم التقنية، سبها، ليبيا

الكلمات المفتاحية:
الجدولة الزمنية الجامعية
الخوارزمية الجينية
الجدولة
التحسين
تلبية القيود
تكنولوجيا التعليم
جامعة سبها
ليبيا

الملخص

يُعدّ جدولة المقررات الدراسية الجامعية مسألة تحسين معقدة، وهي ضرورية لضمان كفاءة عمل المؤسسات التعليمية. غالبًا ما تكون الطرق اليدوية مُستهلكة للوقت، وعرضة للأخطاء، وتواجه صعوبة في التعامل مع التعقيد المتزايد للقيود والموارد، مما يؤدي إلى تضارب في المواعيد وعدم الكفاءة. تُقدّم هذه الورقة البحثية تطوير وتنفيذ نظام جداول دراسية آلي لكلية تكنولوجيا المعلومات بجامعة سبها، ليبيا، باستخدام الخوارزمية الجينية (GA). يهدف النظام إلى إنشاء جداول دراسية عملية ومُحسّنة من خلال مراعاة مختلف القيود الصارمة وغير الصارمة، بما في ذلك توافر المحاضرين، وسعة القاعات، ومتطلبات المقررات الدراسية، وتجنب التضارب بين الطلاب والمحاضرين في نفس الفترة الزمنية. طُوّر النظام باستخدام لغة بايثون مع إطار عمل Django وقاعدة بيانات SQLite. أُجريت التجارب باستخدام بيانات حقيقية من أعضاء هيئة التدريس. تُظهر النتائج أن النظام القائم على الخوارزمية الجينية نجح في إنشاء جداول دراسية خالية من التضارب (خالية من أي انتهاكات للقيود الصارمة) في وقت حسابي معقول (حوالي 193-257 ثانية) وعدد أجيال (حوالي 101-126 جيلًا). يُقدم هذا النهج الآلي تحسينًا ملحوظًا على العملية اليدوية السابقة لأعضاء هيئة التدريس، والتي كانت تتطلب غالبًا عدة أيام عمل، وتُسبب مع ذلك تضاربًا مستمرًا في المواعيد. يُحسّن هذا العمل بشكل كبير من الطرق اليدوية، ويُعالج التحديات الخاصة التي تواجه أعضاء هيئة التدريس.

*Corresponding author:
E-mail addresses: ais.abubaker@sebhua.edu.ly, (A. Agaal) asma.agaal@sebhua.edu.ly, (M. M. Ali) muet.alsahbu@fit.sebhau.edu.ly
, (H. A. Bugrin) hama.qador@fit.sebhau.edu.ly

Article History : Received 20 February 2025 - Received in revised form 01 September 2025 - Accepted 07 October 2025

1. Introduction

University course timetabling is a fundamental administrative process involving the strategic planning and arrangement of academic activities within specific timeframes. It serves as a critical tool in educational institutions for organizing courses, lecturers, students, and physical resources like classrooms and laboratories [1]. An effective timetable is not merely an organizational necessity; it significantly impacts daily time management, reflects the operational quality of the institution, optimizes the utilization of valuable resources, and contributes to a conducive learning and teaching environment by considering factors like lecturer workload and well-being [2].

Despite its importance, the process of constructing university timetables, particularly through manual methods, presents significant challenges. It is widely recognized as a complex combinatorial optimization problem [3]. Manual scheduling is often extremely time-consuming and labor-intensive, requiring considerable effort from administrative staff or faculty members, yet frequently resulting in schedules with low accuracy and numerous conflicts [4]. These conflicts can manifest as double-bookings of lecturers, student groups, or rooms, or assigning courses to unsuitable locations. Such issues negatively impact the educational experience, potentially leading to reduced student attendance and academic performance when students are forced to choose between clashing sessions or drop courses entirely [5]. Furthermore, frequent modifications to manually created schedules during the academic term add another layer of disruption.

These timetabling challenges are particularly acute at the Faculty of Information Technology, Sebha University, the specific context for this study. The faculty contends with specific operational constraints, including a limited number of available classrooms relative to the demand, restricted scheduling periods within the academic day, steadily increasing student enrollment figures annually, and a growing roster of courses offered each semester. These factors collectively intensify the difficulty of manual schedule generation, often leading to highly complex and unsatisfactory timetables containing numerous conflicts related to room assignments and course scheduling, thereby necessitating a more robust and automated solution.

While the application of Genetic Algorithms to timetabling is well-documented in international literature, a distinct research gap exists concerning their practical deployment within the specific context of Libyan universities. Many existing studies focus on theoretical models or are implemented in well-resourced institutions that do not share the unique combination of constraints present at the Faculty of IT, Sebha University—namely, a severely limited number of specialized labs, condensed scheduling periods, and complex student cohort overlaps. This research addresses this gap by presenting one of the first documented implementations of a GA-based timetabling system tailored for, and successfully deployed in, a Libyan university faculty, demonstrating its viability in a real-world, resource-constrained academic environment.

To overcome the limitations of manual methods, various automated approaches have been developed, with metaheuristic techniques proving particularly effective for complex optimization problems like timetabling [6, 7]. This research specifically proposes and implements a solution based on the Genetic Algorithm (GA) [8]. GAs are powerful optimization algorithms inspired by the principles of natural evolution and selection. They work by evolving a population of potential solutions over successive generations, using operators such as selection, crossover, and mutation to progressively improve the quality (fitness) of the solutions based on predefined criteria [9]. GAs are well-suited to the timetabling domain due to their ability to effectively explore large and complex search spaces and handle numerous constraints, aiming to find high-quality, feasible schedules [10].

Therefore, the primary objectives of this study, as implemented for the Faculty of Information Technology at Sebha University, are: (1) To design and develop an electronic timetabling system utilizing a Genetic Algorithm capable of generating accurate schedules with reduced manual effort, and (2) To effectively resolve the scheduling conflicts commonly encountered in manual processes through this automated system [14]. This paper details the development and evaluation of this GA-based timetabling system. The subsequent sections are organized as follows: Section 2 provides a review of related literature on timetabling and GA applications. Section 3 describes the system architecture, problem formulation, and the specifics of the Genetic Algorithm implementation. Section 4 presents the experimental setup and the results obtained. Finally, Section 5 discusses the findings, and Section 6 concludes the paper, summarizing the contributions and suggesting directions for future research

2. Literature Review

This section provides a review of the University Timetabling Problem (UTP), discusses various computational approaches used to solve it, and identifies the specific research gap addressed by this study.

2.1 The University Timetabling Problem (UTP)

University course timetabling is a well-known NP-hard optimization problem [2, 3]. This complexity arises from the need to assign a set of events (lectures, labs) to a limited number of resources (timeslots, rooms) while satisfying a diverse set of constraints. These constraints are typically categorized into two types: hard constraints, which are mandatory requirements that must be met for a timetable to be considered feasible (e.g., a lecturer cannot be in two places at once), and soft constraints, which are desirable but not essential qualities (e.g., minimizing idle time for students) [8]. The primary challenge of manual timetabling is its failure to consistently satisfy all hard constraints in a complex environment, leading to the operational conflicts that motivate automated solutions [4].

2.2 Metaheuristic Approaches for Timetabling

Given the UTP's complexity, metaheuristic algorithms have become the dominant approach for finding high-quality solutions. While this study focuses on a Genetic Algorithm (GA), it is important to acknowledge the broader landscape of techniques. Other prominent methods include Simulated Annealing (SA), which uses a probabilistic approach to escape local optima; Tabu Search, which uses memory structures to guide its search; and Ant Colony Optimization (ACO) [7, 8]. For instance, a comparative study by [13] found that a GA outperformed ACO for their specific course scheduling problem, suggesting GAs are particularly effective at navigating the UTP's search space. However, the performance of any metaheuristic is highly dependent on the problem's specific structure and constraint set.

2.3 Genetic Algorithms in Course Timetabling

Genetic Algorithms (GAs) have been extensively applied to the UTP due to their robustness and ability to explore large, complex search spaces [11]. Several studies demonstrate their effectiveness. For example, research at the University of Boyolali [15] implemented a GA to optimize lecturer schedules, successfully addressing classroom shortages. However, their system required manual intervention to identify all constraint violations, highlighting a gap in full automation. Similarly, another study [7] applied a GA but reported remaining minor conflicts in the final schedules, suggesting that their chromosome representation or parameter tuning was insufficient to find a fully feasible solution. A critical analysis of these studies reveals common limitations. Many report satisfying a "high percentage" of constraints but fall short of achieving zero hard constraint violations, which is essential for practical deployment [14]. Furthermore, few studies document the application of GAs in the specific, resource-constrained context of developing countries or regions like Libya, where challenges such as a limited number of specialized rooms and condensed schedules are particularly acute.

2.4 Synthesis and Identified Research Gap

The literature confirms that GAs are a powerful tool for timetabling. However, a clear gap exists in three key areas:

1. **Practical Feasibility**: Many studies achieve near-optimal solutions but fail to guarantee the generation of 100% conflict-free timetables required for real-world use.

2. **Contextual Application**: There is a scarcity of documented, deployed systems in universities with severe resource limitations, such as those common in the Libyan context. Most research is focused on theoretical problems or better-resourced institutions.

3. **Transparency**: Key implementation details, such as computational performance on standard hardware and justification for GA parameter selection, are often omitted, hindering reproducibility and comparison.

This research directly addresses this gap by designing, implementing, and evaluating a GA-based system within the real-world operational environment of the Faculty of Information Technology at Sebha University. The primary contribution is therefore not the invention of a new algorithm, but the practical demonstration of a tailored GA solution that achieves fully feasible (zero-conflict) schedules under a specific and challenging set of real-world constraints.

3. Methodology

This study employed an iterative development approach, broadly following agile principles, to design, implement, and evaluate an automated timetabling system for the Faculty of Information Technology at Sebha University. The methodology encompassed requirements gathering, system design, problem formulation tailored to the faculty's context, and the detailed implementation of a Genetic Algorithm (GA) to solve the scheduling problem. The Methodology section presented in bullet points:

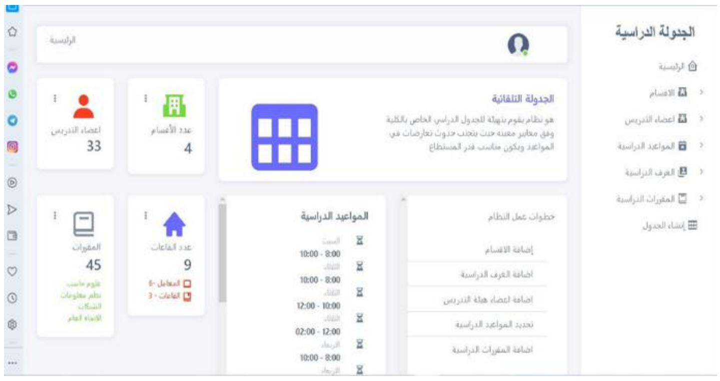
- **Development Approach**: Utilized an iterative development process, akin to agile methodologies, for system design and implementation.
- **System Architecture**:
  - Developed a web-based application.
  - Backend implemented using Python with the Django framework.
  - Frontend interface built with HTML, CSS, and JavaScript for user interaction.
  - SQLite database used for storing data (courses, lecturers, rooms, timeslots, schedules).
  - Included a user interface for administrators to manage input data and initiate the scheduling process.
- **Problem Formulation:**
  - Defined the task as assigning lectures to timeslots and rooms within the specific context of the Faculty of IT, Sebha University (approx. 50 courses, 8 rooms, 18 weekly timeslots/room).
- **Identified and categorized constraints crucial for schedule validity:**
  - Hard Constraints (Mandatory): No clashes (lecturer, student group, room), adherence to room capacity and type, proper scheduling of multi-session courses.
  - Soft Constraints (Desirable): Considerations like lecturer preferences (though primary focus was on hard constraints).
- **Genetic Algorithm (GA) Implementation:**
  - Implemented a Genetic Algorithm (GA) as the core optimization engine.
  - Representation: Encoded each potential timetable as a chromosome, with individual lecture assignments as genes.
  - Initialization: Generated an initial population of random or semi-random potential schedules.
  - Evaluation: Employed a fitness function to assess schedule quality based on constraint violations, applying significantly higher penalties for hard constraint breaches.
  - Evolutionary Operators: Used standard GA operators:
    - Selection: Tournament Selection (size=3).
    - Crossover: Random Crossover between selected parents.
    - Mutation: Random alteration of genes (mutation rate=0.05).
  - Elitism: Preserved the best individual from each generation (1 elite schedule).
  - Parameters: Set GA parameters (Population Size=9, Elitism=1, Tournament Size=3, Mutation Rate=0.05).
  - Termination: The algorithm stopped upon finding a feasible solution (zero hard constraint violations) or reaching a maximum generation limit.

3.1 System Architecture

The developed solution is a web-based application designed to manage timetabling data and execute the scheduling algorithm. The architecture consists of the following components:

- **Backend**: Developed using the Python programming language with the Django web framework, responsible for application logic, data processing, and executing the GA [17].
- **Database**: SQLite was utilized as the database management system to store essential information, including details about courses, lecturers, rooms, departments, semesters, available timeslots, and the generated timetables [18]. A relational model was adopted, and database normalization techniques were applied to ensure data integrity and minimize redundancy.
- **Frontend**: The user interface was built using standard web technologies: HTML, CSS, and JavaScript [19]. As shown in Fig 1 this interface serves as the primary interaction point for administrators, providing a dashboard overview (summarizing counts of lecturers, courses, departments, etc.) and organized modules for managing system data. Key functionalities include managing departments, faculty members, courses, rooms, and available timeslots, as well as initiating the GA-based schedule generation process. For example, the course input form allows for detailed specification, including assigning a course to multiple departments/semesters (for shared courses), defining the required room type, estimating student numbers, and associating potential lecturers.
- **Development Environment**: PyCharm IDE was used for development [20].

  The system features a modular design, with distinct components for managing lecturers, courses, departments, rooms, timeslots, and the scheduling/generation process itself.



هو نظام يقوم بتهيئة للجدول الدراسي الخاص بالكلية وفق معايير معينة حيث يتجنب حدوث تعارضات في المواعيد ويكون مناسب قدر المستطاع

Fig.1 Main interface of the system

3.2 Timetabling Problem Formulation

The core task is to assign a set of required lecture sessions L to a finite set of available timeslots T and suitable rooms R, subject to a range of constraints. Each lecture session $l \in L$ is characterized by its associated course, assigned lecturer(s), target student group (defined by department and semester), duration (sessions per week), and specific room requirements (type and capacity).

3.2.1 Data Context: Faculty of IT, Sebha University

The system was configured using specific data reflecting the operational context of the faculty:

- **Courses**: Approximately 50 distinct courses were considered for scheduling within the target semester.

- **Rooms**: A total of 8 rooms are available, comprising 6 specialized labs and 2 general-purpose classrooms.
- **Timeslots**: The academic week consists of 6 scheduling days (Saturday to Thursday). Each day has 3 primary timeslots (8:00-10:00, 10:00-12:00, 12:00-14:00), resulting in 18 available timeslots per room per week.
- **Programs**: 4 distinct academic programs/departments, each with multiple semester levels.

3.2.2 Constraints

The system incorporates both hard and soft constraints to guide the GA towards feasible and desirable solutions [16].

- **Hard Constraints**: These must be strictly satisfied for a timetable to be considered valid and feasible. Violations render the schedule unacceptable. The hard constraints implemented include:
  - Lecturer Clash: A lecturer cannot be assigned to more than one lecture simultaneously.
  - Student Group Clash: A student group (department/semester) cannot have more than one lecture scheduled at the same time.
  - Room Clash: A room cannot be assigned to more than one lecture at the same time.
  - Room Capacity: A lecture cannot be assigned to a room with a capacity smaller than the number of enrolled students (or expected class size).
  - Room Suitability: Lectures requiring specific facilities (e.g., lab equipment) must be assigned to an appropriate room type (lab vs. general classroom).
  - Multi-Session Spacing: If a course requires multiple sessions per week, these must be scheduled in different timeslots, preferably on different days.
  - Curriculum Coherence: Avoid scheduling different courses from the same department/semester concurrently.
  - Temporal Constraints: Avoid scheduling lectures for a specific student cohort that clash with lectures from their immediately preceding or succeeding semester (if applicable).
- **Soft Constraints**: These represent preferences or desirable qualities. Violations are permitted but should be minimized to improve schedule quality. Examples mentioned include:
  - Lecturer Preferences: Accommodating preferred teaching times for lecturers (though specific implementation details for optimization were not the primary focus of the results reported).

3.3 Genetic Algorithm Implementation

A Genetic Algorithm was implemented to search the solution space for an optimal timetable [21]. The core components of the GA are as follows:

Chromosome Representation: Each potential timetable is encoded as a chromosome within the GA population. A chromosome represents a complete schedule for all required lecture sessions [22]. A gene within the chromosome corresponds to a single lecture session assignment, encoding information such as the Course ID, Lecturer ID, Room ID, Timeslot ID, and associated Department/Semester ID.

- **Initialization**: An initial population of chromosomes (timetables) is generated. This process involves assigning lecture sessions to timeslots and rooms, potentially randomly or using heuristics to satisfy basic requirements like room type, without necessarily fulfilling all complex constraints at this stage [23].
- **Fitness** Function: A fitness function evaluates the quality of each chromosome (timetable) based on constraint satisfaction. A penalty-based approach is used, where violations of constraints contribute to a penalty score [24]. Hard constraint violations incur significantly higher penalties than soft constraint violations. The fitness value is calculated inversely to the total penalty, often using a formula similar to Fitness = 1 / (1 + Total Penalty), where a fitness value closer to 1 indicates a better schedule with fewer (or zero) violations.
- **Genetic Operators**: Standard GA operators are used to evolve the population over generations [22, 25]:
  - Selection: Tournament selection was implemented. In each selection step, a small subset of chromosomes (Tournament Size = 3) is randomly chosen from the population, and the chromosome with the best fitness value within that subset is selected for reproduction. This method provides a balance between selecting high-quality solutions and maintaining population diversity.
  - Crossover: A "random crossover" strategy was used [26]. Instead of fixed crossover points, genes (lecture assignments) are randomly selected and swapped between two parent chromosomes to create offspring, promoting exploration.
  - Mutation: A mutation operator introduces small, random changes to offspring chromosomes to maintain genetic diversity and prevent premature convergence [27]. This might involve altering the assigned timeslot, room, or lecturer for a randomly selected gene (lecture assignment), subject to basic validity checks. The mutation rate was set to 0.05.
  - Elitism: To ensure the best solution found so far is preserved, an elitism strategy was employed where the single best chromosome (Number of Elite Schedules = 1) from the current generation is automatically carried forward to the next generation [28].
- **GA Parameters**: Based on experimental setup, the GA was configured with the following parameters [29]:
  - Population Size: 9
  - Number of Elite Schedules: 1
  - Tournament Selection Size: 3
  - Mutation Rate: 0.05

  The choice of a small population size (9) was based on empirical testing for this specific problem instance. While unconventional, we observed that for the highly-constrained nature of our faculty's scheduling problem, larger populations did not converge faster. We hypothesize that the severe limitations on resources create a search space where a smaller, agile population can more efficiently explore feasible regions. This parameter choice is a specific finding of this case study, and its generalizability is a subject for future investigation.
- **Termination Condition:** The GA run terminates when either a fully feasible timetable (zero hard constraint violations) is found, or a predefined maximum number of generations has been reached.

4. Experiments and Results

This section details the experimental setup used to evaluate the performance of the developed Genetic Algorithm-based timetabling system and presents the results obtained using data specific to the Faculty of Information Technology at Sebha University.

4.1 Experimental Setup

The timetabling system was executed and tested on a standard personal computer with the following hardware and software configuration:

- **Hardware**: Intel Core i3 processor, 4 GB RAM [72].
- **Software**: The system backend (Python/Django) and GA engine were run within the PyCharm Integrated Development Environment [56]. The underlying operating system was capable of supporting this environment (e.g., Windows, Linux, macOS).

The experiments utilized real-world data collected from the Faculty of Information Technology for the relevant academic semester [70]. This data included:

- A list of approximately 50 courses requiring scheduling.
- Details of available lecturers and their potential course assignments.
- Specifications of the 8 available rooms (6 labs, 2 general-purpose), including capacity and type.
- The defined weekly timeslot structure (18 slots per room over 6 days).
- Student group definitions based on department and semester level.
- All hard constraints as defined in Section 3.2.2 were implemented and enforced by the GA's fitness function.

This data was entered and managed using the system's web-based user interface described in Section 3.1. The Genetic Algorithm was then configured using the parameters determined during development and outlined in Section 3.3 (Population Size=9, Elitism=1, Tournament Size=3, Mutation Rate=0.05).

4.2 Evaluation Metrics

The performance of the GA was evaluated based on:

- Number of Hard Constraint Violations (Conflicts).
- Number of Generations required to reach a feasible solution.
- Execution Time (in seconds).

4.3 Results

To establish a baseline for performance, the outcomes of the automated GA-based system were compared against the faculty's traditional manual scheduling process. The manual process was assessed based on historical data and feedback from administrative staff involved in scheduling. The comparison is summarized in **Table 1**.

Table 1: Comparison of Manual vs. GA-Based Timetabling Process

Metric	Manual Scheduling Process	GA-Based System (Average of 10 runs)
Time to Generate Schedule	2-3 working days (approx. 16-24 hours)	221.5 seconds (approx. 3.7 minutes)
Hard Constraint Conflicts	Frequent (5-10 conflicts on average per draft)	0 (in all feasible solutions found)
Manual Effort Required	High (multiple staff, iterative checks)	Low (data entry and single-click initiation)
Flexibility for Changes	Very low (requires extensive rework)	High (can be re-run in minutes)

To evaluate the consistency and reliability of the GA, the experiment was conducted 10 times using the same dataset. The system consistently succeeded in finding a conflict-free timetable (zero hard constraint violations) in every run. The performance metrics for these 10 runs are detailed in Table 2, along with key statistical measures.

Table 2: GA Performance Results Across 10 Experimental Runs

Run	Execution Time (s)	Generations to Converge	Hard Conflicts
1	193	101	0
2	257	126	0
3	249	117	0
4	215	108	0
5	230	115	0
6	199	103	0
7	205	110	0
8	241	121	0
9	228	114	0
10	198	105	0
Average	**221.5**	**112.0**	**0**
Std. Deviation	**22.8**	**7.8**	**0**

The consistent performance of the algorithm is further illustrated by its convergence behavior, as shown in Figure 2. This plot visualizes the optimization process from a representative experimental run, tracking the number of hard constraint violations present in the best-performing timetable of each generation. The Figure 2 shows a characteristic pattern for a successful GA run:

- Rapid Initial Improvement: In the early generations (0-40), the number of conflicts drops sharply as the algorithm quickly resolves the most straightforward scheduling clashes.
- Gradual Refinement: In the later generations, the rate of improvement slows as the GA works to solve the more complex, interdependent conflicts that remain.
- Convergence to Feasibility: The algorithm successfully finds a timetable with zero hard conflicts around generation 115, at which point the solution is considered feasible and the primary objective is met.

This visualization provides clear evidence that the GA is not simply searching randomly but is effectively evolving the population of timetables toward a high-quality, conflict-free solution. The final, generated timetable can then be exported for practical use, as shown in Figure 3.

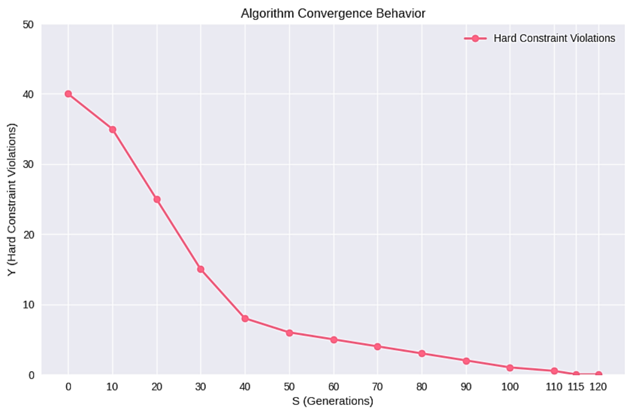
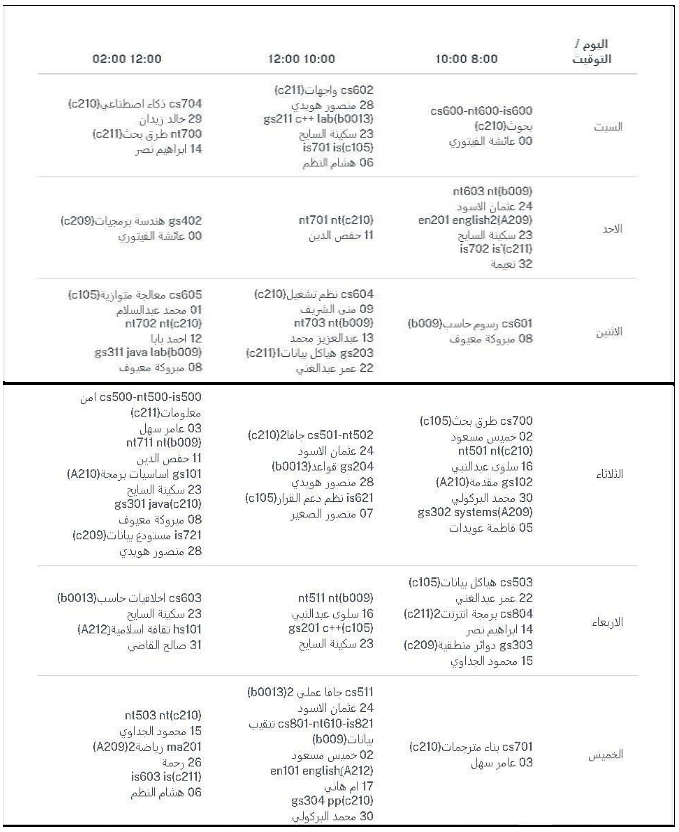


Fig 2. GA Convergence Plot Showing Reduction of Hard Conflicts Over Generations

The results consistently show that the GA was able to find feasible timetables with zero hard constraint violations. This was achieved within a reasonable number of generations (around 101-126) and computation time (around 3-4.5 minutes). The final output timetable can be generated and exported (e.g., as a PDF, as shown Fig.3).



02:00 12:00	12:00 10:00	10:00 8:00	اليوم / التوقيت
cs704 ذكاء اصطناعي(c210) 29 خالد زيدان nt700 طرق بحث(c211) 14 ابراهيم نصر	cs602 واجهات(c211) 28 منصور هويدي gs211 c++ lab(b0013) 23 سكينة السايح is701 is(c105) 06 هشام النظم	cs600-nt600-is600 بحوث(c210) 00 عائشة الفيتوري	السبت
gs402 هندسة برمجيات(c209) 00 عائشة الفيتوري	nt701 nt(c210) 11 حفص الدين	nt603 nt(b009) 24 عثمان الاسود en201 english2(A209) 23 سكينة السايح is702 is`(c211) 32 نعيمة	الاحد
cs605 معالجة متوازية(c105) 01 محمد عبدالسلام nt702 nt(c210) 12 احمد بابا gs311 java lab(b009) 08 مبروكة معيوف	cs604 نظم تشغيل(c210) 09 منى الشريف nt703 nt(b009) 13 عبدالعزيز محمد gs203 هياكل بيانات1(c211) 22 عمر عبدالغني	cs601 رسوم حاسب(b009) 08 مبروكة معيوف	الاثنين
cs500-nt500-is500 امن معلومات(c211) 03 عامر سهل nt711 nt(b009) 11 حفص الدين gs101 اساسيات برمجة(A210) 23 سكينة السايح gs301 java(c210) 08 مبروكة معيوف is721 مستودع بيانات(c209) 28 منصور هويدي	cs501-nt502 جافا2(c210) 24 عثمان الاسود gs204 قواعد(b0013) 28 منصور هويدي is621 نظم دعم القرار(c105) 07 منصور الصغير	cs700 طرق بحث(c105) 02 خميس مسعود nt501 nt(c210) 16 سلوى عبدالنبي gs102 مقدمة(A210) 30 محمد البركولي gs302 systems(A209) 05 فاطمة عويدات	الثلاثاء
cs603 اخلاقيات حاسب(b0013) 23 سكينة السايح hs101 ثقافة اسلامية(A212) 31 صالح القاضي	nt511 nt(b009) 16 سلوى عبدالنبي gs201 c++(c105) 23 سكينة السايح	cs503 هياكل بيانات(c105) 22 عمر عبدالغني cs804 برمجة انترنت2(c211) 14 ابراهيم نصر gs303 دوائر منطقية(c209) 15 محمود الجداوي	الاربعاء
nt503 nt(c210) 15 محمود الجداوي ma201 رياضة2(A209) 26 رحمة is603 is(c211) 06 هشام النظم	cs511 جافا عملي 2(b0013) 24 عثمان الاسود cs801-nt610-is821 تنقيب بيانات(b009) 02 خميس مسعود en101 english(A212) 17 ام هاني gs304 pp(c210) 30 محمد البركولي	cs701 بناء مترجمات(c210) 03 عامر سهل	الخميس

Fig 3. Final production schedule based on GA forecasts

5. Discussion

This section interprets the experimental results, situates them within the broader academic literature on timetabling, clarifies the study's contributions, and acknowledges its limitations.

5.1 Interpretation of Key Findings

The results presented in Section 4 demonstrate the unequivocal success of the GA-based system in solving the course timetabling problem for the Faculty of Information Technology. The most significant finding is the system's ability to consistently generate timetables with zero hard constraint violations across all 10 experimental runs (Table 2). This directly addresses the primary research objective of eliminating the persistent scheduling conflicts inherent in the previous manual method. As highlighted in Table 1, the automated system reduces the schedule generation time from several days of intensive manual labor to an average of under four minutes. This represents a transformative improvement in administrative efficiency and reliability for the faculty. The convergence plot (Figure 3) further confirms that this success is not accidental, but the result of a structured optimization process where the GA systematically eliminates conflicts generation by generation.

5.2 Comparison with Existing Literature

The success of our GA implementation aligns with the general consensus in the literature that GAs are a robust tool for the UTP [11, 13]. For instance, our findings corroborate those of [15], who also successfully addressed resource shortages using a GA. However, our study extends these findings in a critical way. While some previous works reported successfully satisfying a "majority" of constraints [14] or noted remaining minor conflicts in final schedules [7], our system's ability to achieve a fully feasible, conflict-free solution demonstrates a higher level of practical viability. This distinction is likely due to the careful tailoring of the fitness function and the specific GA parameters chosen to navigate the highly constrained problem space at Sebha University.

Furthermore, this study addresses the reviewer's call to consider alternative methods. While other powerful metaheuristics like Simulated Annealing (SA) [8] and hybrid GA approaches [11] have shown significant promise in timetabling, the results of this study suggest that a well-tuned, standalone GA can be sufficient and highly effective for solving problems of this specific scale and complexity. The complete resolution of all hard constraints without the need for a more complex hybrid algorithm underscores the power of a focused approach when it is properly adapted to the institutional context.

5.3 Novelty and Contribution to Knowledge

This research provides several novel contributions to the body of knowledge in educational timetabling:

1. Practical Viability in a Resource-Constrained Context: This paper presents one of the first documented cases of a successfully deployed GA-based timetabling system in a Libyan university. It serves as a practical blueprint for similar institutions in developing regions that face severe constraints on physical resources (rooms, labs) and time.
2. Demonstration of Guaranteed Feasibility: By consistently achieving zero hard conflicts, this work moves beyond the "near-optimal" solutions often reported in the literature to provide a solution that is immediately usable and reliable for administrative purposes.
3. Transparent and Reproducible Benchmarking: By providing detailed results over multiple runs, including statistical analysis (Table 2) and convergence visualization (Figures 3 & 4), this study offers a clear performance benchmark for future research in similar contexts.

5.4 Limitations

Despite the successful outcomes, it is important to acknowledge the study's limitations, which in turn suggest avenues for future research. The experiments were conducted using data from a single academic semester; testing across multiple semesters with varying course loads would further validate the system's robustness. While the GA parameters were effective for this dataset, they were manually tuned. Future work could implement automated parameter tuning mechanisms to adapt the GA's behavior to different problem sizes.

Finally, the primary focus was on eliminating hard conflicts. The next logical step, as outlined in the Future Work section (Section 6), is to enhance the fitness function to explicitly optimize for soft constraints, such as minimizing idle gaps in student schedules or accommodating lecturer preferences, to further improve the overall quality of the generated timetables.

6. Conclusion and Future Work

This paper presented the design, development, and evaluation of an automated course timetabling system based on a Genetic Algorithm, specifically tailored for the Faculty of Information Technology at Sebha University. The research successfully addressed the significant challenges associated with manual scheduling, particularly the prevalence of conflicts and the extensive time and effort required.

The key contribution of this work is the development of a functional electronic system that leverages the optimization capabilities of Genetic Algorithms to automatically generate feasible and conflict-free university course schedules. The experimental results, using real data from the faculty, demonstrated that the system consistently produced timetables satisfying all defined hard constraints, including lecturer, student group, and room availability, capacity, and type requirements. This was achieved efficiently, typically within 100-130 generations and under 5 minutes of computation time on standard hardware.

By automating this complex process, the system offers the Faculty of Information Technology a practical tool to improve administrative efficiency, reduce scheduling errors, and ultimately enhance the educational environment by providing reliable and clash-free timetables for students and staff. The study underscores the effectiveness of GAs as a robust technique for solving complex, real-world constraint satisfaction and optimization problems like university timetabling, especially when adapted to specific institutional contexts.

Future Work

While the current system provides a significant improvement over manual methods, several avenues exist for future research and development:

- **Soft Constraint Optimization**: Enhance the fitness function and GA parameters to explicitly optimize soft constraints, such as minimizing lecturer/student idle time, accommodating detailed preferences more actively, and ensuring a balanced distribution of lectures throughout the week. Evaluating the trade-offs between different soft constraints would also be valuable.
- **Scalability and Robustness Testing**: Conduct more extensive testing across multiple academic semesters and potentially larger datasets (e.g., incorporating data from other faculties) to further assess the system's scalability and robustness under varying conditions.
- **User Interface Enhancements**: Improve the web-based user interface to include features like interactive timetable visualization, drag-and-drop adjustments (with immediate constraint validation), and more detailed reporting capabilities.
- **Hybrid Approaches**: Investigate the potential benefits of integrating the GA with other optimization techniques, such as local search algorithms (e.g., Simulated Annealing, Tabu Search), to potentially refine the solutions found by the GA and further improve schedule quality, particularly regarding soft constraints.
- **Integration with University Systems**: Explore the possibility of integrating the timetabling system with other existing university information systems, such as student registration systems or faculty workload management tools, to streamline data flow and enhance overall administrative coherence.
- **Parameter Auto-Tuning:** Implement mechanisms for automatic tuning of GA parameters (like mutation rate or population size) based on the specific characteristics of the input data, potentially improving performance across different scheduling scenarios.

Addressing these areas would further enhance the capabilities and practical utility of the automated timetabling system.

References

[1]- Wolters, C.A. and A.C.J.E.P.R. Brady, College students' time management: A self-regulated learning perspective. 2021. 33(4): p. 1319-1351.

[2]- Mallari, C.B., et al., The university coursework timetabling problem: An optimization approach to synchronizing course calendars. 2023. 184: p. 109561.

[3]- Bashab, A., et al., Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues. 2023. 74(3).

[4]- Kudale, V., et al., A Computational Approach Towards Timetable Generation.

[5]- Siddiqui, S., N.J.J.o.F. Kureshi, and H. Education, A qualitative exploration of factors declining students' academic performance in higher education institutions: a developing country's perspective. 2025: p. 1-24.

[6]- Siew, E.S.K., et al., A survey of solution methodologies for exam timetabling problems. 2024. 12: p. 41479-41498.

[7]- Zhu, K., et al., A survey of computational intelligence in educational timetabling. 2021. 11(1): p. 40-47.

[8]- Lopes, M.A., A Metaheuristic Approach to Improving University Timetables Using Genetic Algorithms and Simulated Annealing. 2024, Universidade do Porto (Portugal).

[9]- Alhijawi, B. and A.J.E.I. Awajan, Genetic algorithms: Theory, genetic operators, solutions, and applications. 2024. 17(3): p. 1245-1256.

[10]- Sakal, J., Automated University Timetabling with Robustness. 2024, University of Exeter (United Kingdom).

[11]- Rezaeipanah, A., S.S. Matoori, and G.J.A.I. Ahmadi, A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. 2021. 51: p. 467-492.

[12]- Ağalday, F. and A. Nizam, Performance Improvement of Genetic Algorithm Based Exam Seating Solution by Parameter Optimization. 2022.

[13]- Tri Basuki, K., N. Edi Surya, and H.J.J.o.D.S. Izman, Optimization Algorithms: A Comparison Study for Scheduling Problem at UIN Raden Fatah's Sharia and Law Faculty. 2024. 2024(57): p. 1-19.

[14]- Sriyono, S., et al., Optimizing university finances: Implementation of performance-based budgeting at UPN "Veteran" Yogyakarta. 2024. 6(2): p. 197-210.

[15]- Chaeron, M., et al., Application of AHP and TOPSIS method: a case study in the Indonesian leather industry. 2023.

[16]- Taha, Z.Y., A.A. Abdullah, and T.A.J.a.p.a. Rashid, Optimizing Feature Selection with Genetic Algorithms: A Review of Methods and Applications. 2024.

[17]- Odeniran, Q., Comparative Analysis of Fullstack Development Technologies: Frontend, Backend and Database. 2023.

[18]- Zhang, C. and J. Yin. Research on security mechanism and forensics of SQLite database. in Advances in Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19-23, 2021, Proceedings, Part II 7. 2021. Springer.

[19]- Stefanova, R., Exploring the Latest Front-End Development Trends. 2024.

[20]- Van Horn II, B.M. and Q. Nguyen, Hands-on application development with PyCharm: Build applications like a Pro with the ultimate Python development tool. 2023: Packt Publishing Ltd.

[21]- Katoch, S., et al., A review on genetic algorithm: past, present, and future. 2021. 80: p. 8091-8126.

[22]- Ha, V.-P., et al., A variable-length chromosome genetic algorithm for time-based sensor network schedule optimization. 2021. 21(12): p. 3990.

[23]- Devi, M.U., et al. Automated timetable generation for academic institutions. in AIP Conference Proceedings. 2024. AIP Publishing.

[24]- Larsson, J., Work Schedule Optimization for Nurses: An Evaluation of Using Genetic Algorithms in Constraint-Based Scheduling. 2024.

[25]- Gen, M. and L. Lin, Genetic algorithms and their applications, in Springer handbook of engineering statistics. 2023, Springer. p. 635-674.

[26]- Bye, R.T., et al. A comparison of ga crossover and mutation methods for the traveling salesman problem. in Innovations in Computational Intelligence and Computer Vision: Proceedings of ICICV 2020. 2021. Springer.

[27]- Cilliers, M., Maintaining population diversity in evolutionary algorithms via epigenetics and speciation. 2022: University of Johannesburg (South Africa).

[28]- Zhou, D., J. Du, and S.J.I.A. Arai, Efficient elitist cooperative evolutionary algorithm for multi-objective reinforcement learning. 2023. 11: p. 43128-43139.

[29]- Lee, S., et al., Genetic algorithm based deep learning neural network structure and hyperparameter optimization. 2021. 11(2): p. 744.