Volume 11 Issue 1 Year 2026 | Page 381-392 ISSN: 2527-9866
Received: ***03-01-2026*** | *Revised:* ***20-01-2026*** | *Accepted:* ***26-02-2026***

Application of Genetic Algorithm and Or-Tools for Cloud-Based Course Scheduling Optimization

Salamul Jabbar[1], Safwandi[2], Kurniawati[3], Eva Darnila[4], Wahyu Fuadi[5]
[1,2,34,5)] Malikussaleh University, Muara Batu, North Aceh Regency, Aceh
e-mail: salamul.210170292@mhs.unimal.ac.id[1)], safwandi@unimal.ac.id[2)], kurniawati@unimal.ac.id[3)]
eva.darnila@unimal.ac.id[4)], wahyu.fuadi@unimal.ac.id[)]
*Correspondence: salamul.210170292@mhs.unimal.ac.id

Abstract: *Course scheduling in higher education institutions is a complex combinatorial optimization problem involving numerous constraints such as lecturer availability, room capacity, time slots, and course distribution across semesters. Manual scheduling practices often result in conflicts, inefficient resource utilization, and prolonged preparation time. This study proposes a hybrid course scheduling system that integrates a Genetic Algorithm (GA) and Constraint Programming using the CP-SAT solver from OR-Tools. The GA is employed in the first phase to generate optimal course sections based on student enrollment, lecturer workload, and capacity constraints. The best solution produced by the GA is then refined using CP-SAT to generate a conflict-free timetable that satisfies all hard constraints, including lecturer, room, and time conflicts, while also optimizing selected soft constraints. The proposed system is implemented as a web-based application deployed on Microsoft Azure, enabling scalability and accessibility. Experimental results using real academic data demonstrate that the hybrid approach successfully produces feasible schedules with zero conflicts and significantly reduces scheduling time compared to manual methods. The results confirm that the integration of GA and CP-SAT provides an effective and flexible solution for university course scheduling problems.*

Keywords: *course scheduling, genetic algorithm, constraint programming, OR-Tools, optimization*

1. INTRODUCTION

Course scheduling is a critical academic administrative process in higher education institutions. The scheduling process involves assigning courses to time slots, lecturers, and classrooms while satisfying various academic and operational constraints. As the number of courses, students, and lecturers increases, the complexity of the scheduling problem grows exponentially. This problem belongs to the class of NP-hard combinatorial optimization problems, making it difficult to solve optimally using conventional deterministic approaches [1]. In many universities, course scheduling is still performed manually or semi-manually using spreadsheet-based tools. Such approaches are time-consuming, error-prone, and often lead to schedule conflicts such as overlapping lecturer assignments or room usage. In addition, manual scheduling struggles to accommodate soft constraints, including lecturer preferences and balanced workload distribution [2]

Various computational approaches have been proposed to address the course scheduling problem, including Genetic Algorithms (GA), Simulated Annealing, Tabu Search, and Constraint Programming (CP). Genetic Algorithms are particularly effective in exploring large solution spaces and producing near-optimal solutions within reasonable time. However, GA-based solutions may still violate hard constraints if not carefully designed. On the other hand, Constraint Programming guarantees constraint satisfaction but may suffer from scalability issues when dealing with large problem spaces.[2] This research proposes a hybrid scheduling approach that combines the strengths of Genetic Algorithms and Constraint Programming using the CP-SAT solver from OR-Tools. The GA is used for initial section generation and lecturer assignment, while CP-SAT refines the solution

to ensure full constraint satisfaction. The system is implemented as a web-based application deployed on Microsoft Azure, supporting scalability and real-world deployment. The main contribution of this study is the development of a hybrid scheduling framework that produces conflict-free schedules efficiently and is suitable for practical academic environments [3]. This research will implement the system using a cloud computing platform, specifically Azure App Service, to ensure the developed solution is not only algorithmically effective but also reliable and accessible. Scalability, guaranteed by the service provider for high availability, and accessibility to *Microsoft Azure App Service* is *a Platform-as-a-Service* (PaaS) service that makes the deployment and management of web applications easier. With *App Service*, researchers can focus on developing algorithm logic and application features, while Azure automatically handles *server management*, operating systems, and security patching issues. This not only speeds up the development process but also creates a stable and professional environment that supports digital transformation on campus, which is part of Malikussaleh University's commitment to using the latest technology [1].

This research aims to create a lecture scheduling application based on a *hybrid algorithm* (GA and OR-Tools) that can be used as a web-based system. By using this application, the scheduling process can be done automatically [4]. Data such as courses, lecturers, times, and classrooms can be entered, and then processed to create a feasible and optimal schedule that takes preferences into account and eliminates conflicts. This application is expected to solve the classic problem of lecture scheduling by combining intelligent optimization methods, interactive features, and web technology [5]. This condition is very much in line with the real situation in the Informatics Engineering Study Program at Malikussaleh University [6]. According to initial interviews with Study Program Coordinators and administrative staff, the current manual scheduling process takes up to three to seven days each semester. With the number of students increasing every year, Informatics Engineering is a great example of implementing a cloud-based automated system that helps address academic issues and supports digital transformation in the campus environment. This This study proposes a hybrid course scheduling method that integrates a Genetic Algorithm (GA) and Constraint Programming using the CP-SAT solver from Google OR-Tools. The hybrid approach is designed to address the complexity of university course timetabling problems that involve both combinatorial optimization and strict feasibility constraints. The methodology consists of four main stages: data preparation, course section generation using Genetic Algorithm, detailed scheduling using OR-Tools CP-SAT, and schedule evaluation [7].

2. LITERATURE REVIEW

A. System Architecture

The overall system architecture is designed as a modular scheduling pipeline. The input data includes course information, student enrollment, lecturer data, room data, available time slots, and lecturer preferences. These data are stored in structured tables and processed sequentially by the scheduling system [8] . In the first stage, the Genetic Algorithm module processes course and student data to determine the required number of sections for each course and assigns lecturers to these sections. The output of this stage is a list of feasible course sections without time and room assignments [9]. In the second stage, the OR-Tools CP-SAT module receives the section list and performs detailed scheduling by assigning each section to a specific day, time slot, and classroom while enforcing all hard constraints. Soft constraints related to lecturer preferences are incorporated as penalty values. The final output of the system is a complete, conflict-free timetable that satisfies all hard constraints and minimizes preference violations.

B. Algorithm Flow

The proposed scheduling system follows a two-stage hybrid optimization process that integrates a Genetic Algorithm (GA) and OR-Tools CP-SAT. The complete workflow is described as follows:

1. The system initializes the scheduling process and prepares all required academic data for optimization.
2. The system collects master data consisting of courses, lecturers, student enrollment, classroom capacities, available time slots, and lecturer preferences.
3. The Genetic Algorithm determines the number of required sections for each course based on student enrollment and room capacity constraints. At this stage, lecturers are assigned to sections while considering workload balance constraints.
4. The generated sections are evaluated to ensure feasibility, including sufficient capacity, reasonable number of sections, and compliance with basic academic policies. If infeasible, the process is terminated and a failure notification is generated. Feasible sections are forwarded to the CP-SAT solver, which assigns each section to a specific day, time slot, and classroom while enforcing all hard constraints. Hard constraints (lecturer availability, room usage, time conflicts, and capacity) are strictly enforced, while soft constraints (lecturer preferences) are optimized using penalty minimization.
5. The final conflict-free schedule is validated and stored in the system database for visualization and publication. The optimization process ends successfully once a valid timetable is produced.

C. Genetic Algorithm Formulation

The number of required sections for each course is determined based on the total number of enrolled students and the maximum capacity of a classroom or laboratory. This calculation ensures that no section exceeds the available room capacity and that all students can be accommodated:

$$\text{Section Minimal} = \left\lceil \frac{\text{number of students}}{\text{Space Capacity}} \right\rceil$$

This equation guarantees that the total capacity provided by all sections is sufficient to serve the enrolled students, while avoiding overcrowded classes.

Table 1. example of the required section calculation for several courses in the Informatics Engineering program

	Course Name	Semester	Type	Credits (SKS)	Number of Students	Capacity per Section	Required Sections	Total Capacity
1	Algorithm and Programming II	II	Practical	3	259	36	8	288
2	Logic in Informatics	II	Theory	2	259	45	6	270
3	Computer Organization and Architecture	II	Theory	3	259	45	6	270
4	Operating Systems II	II	Theory	3	259	45	6	270
5	Professional Ethics	II	Theory	2	259	45	6	270
6	Statistics and Probability	II	Theory	3	259	45	6	270
7	Digital Fundamentals	II	Theory	3	259	45	6	270
8	Computer Networks	IV	Practical	3	224	36	7	252
9	Web Programming	IV	Practical	3	224	36	7	252
10	Database Management Systems	IV	Practical	3	224	36	7	252
11	Numerical Methods	IV	Theory	3	224	45	5	225

Table I presents an example of the required section calculation for several courses in the Informatics

Engineering program. As shown in the table, courses with a large number of students are divided into multiple parallel sections based on their respective room capacities. For example, the course Algorithm and Programming II has 259 enrolled students and a laboratory capacity of 36 students per section, resulting in 8 required sections with a total capacity of 288 students. This configuration provides a safety margin while ensuring feasibility. Similarly, theory-based courses such as Logic in Informatics and Operating Systems II use a higher room capacity of 45 students per section, reducing the number of required sections to six while still accommodating all enrolled students. Practical courses generally require smaller room capacities due to laboratory constraints, which leads to a higher number of sections compared to theory-based courses with similar enrollment sizes. By incorporating this section calculation mechanism into the Genetic Algorithm, the system ensures that the generated section configuration is feasible from the outset. This approach significantly reduces infeasible solutions in later scheduling stages and improves the overall efficiency of the hybrid GA and CP-SAT optimization process[10].

D. OR-Tools CP-SAT Model

1. Decision Variables

The CP-SAT model uses binary decision variables defined as:

$$x_{c,t,r,l} = \begin{cases} 1, & If\ the\ course\ c\ is\ scheduled\ at\ timeslot\ t, \\ & \quad using\ room\ r\ and\ taught\ by\ lecturer\ l \\ 0, & If\ Not \end{cases}$$

Where:

- *c* represents the section index obtained from the Genetic Algorithm output,
- *t* denotes a time slot defined as a combination of day and start–end time,
- *r* denotes the assigned classroom or online room,
- *d* represents the lecturer assigned to the section.

Each section must be assigned exactly one feasible combination of (*t, r, d)*

2. Hard Constraints

a. Room Constraint

A room cannot be used by more than one section at the same time. This constraint ensures that no overlapping usage of physical or virtual classrooms occurs.

$$\sum_{c,d} X_{c,t,r,d} \leq 1 \quad \forall t, r$$

This formulation fixes a specific time slot tt t and room rr r , then sums all possible sections and lecturers assigned to that pair. The constraint enforces that at most one section occupies the room at that time.

b. **Lecturer Constraint**

Each lecturer can only teach one section at a given time:

$$\sum_{c,d} X_{c,t,r,d} \leq 1 \quad \forall t, d$$

This constraint prevents scheduling conflicts in lecturer assignments and guarantees that a lecturer's teaching load does not overlap across different sections.

c. **Section Uniqueness Constraint**

Each section must be scheduled exactly once:

$$\sum_{c,d} X_{c,t,r,d} \leq 1 \quad \forall c$$

This constraint avoids two invalid conditions: unscheduled sections (sum equals zero) and duplicate schedules for a single section (sum greater than one).

3. Soft Constraints and Penalties

Lecturer preferences are modeled as soft constraints. A penalty is assigned when a scheduled time does not match the lecturer's preferred time:

$$P_c = \begin{cases} 0, & \text{if the schedule is according to preference} \\ 10, & if\ it\ doesn't\ fit \end{cases}$$

Table 2. Lecturer Preferences Model

Lecturer	Time Slot	Room	Preferred Time	Preference Match	Penalty	Fitness
Eva Darnila, ST., MT	Wednesday, 10:30–12:10	On line	Tue: 14:00–16:30 Tue: 10:40–13:10 Tue: 10:30–12:10 Wed: 10:30–12:10 Wed: 14:00–15:40	Excellent	0	1.00
Eva Darnila, ST., MT	Wednesday, 2:00–3:40 PM	On line	Tue: 14:00–16:30 Tue: 10:40–13:10 Tue: 10:30–12:10 Wed: 10:30–12:10 Wed: 14:00–15:40	Excellent	0	1.00
Risawandi, ST., M.Kom	Thursday, 8:00–8:50	On line	–	Not Matched	10	0.09
Risawandi, ST., M.Kom	Thursday, 8:50–9:40 AM	On line	–	Not Matched	10	0.09
Eva Darnila, ST., MT	Tuesday, 10:40–1:10 PM	Info 1	Tue: 14:00–16:30 Tue: 10:40–13:10 Tue: 10:30–12:10 Wed: 10:30–12:10 Wed: 14:00–15:40	Excellent	0	1.00
Eva Darnila, ST., MT	Tuesday, 2:00 PM–4:30 PM	Info 1	Tue: 14:00–16:30 Tue: 10:40–13:10 Tue: 10:30–12:10 Wed: 10:30–12:10 Wed: 14:00–15:40	Excellent	0	1.00

E. Genetic Algorithm Parameters

Table 3. Genetic Algorithm Parameters

Parameter	Value
Chromosome encoding	Course–Lecturer mapping
Population size	100
Number of generations	200
Selection method	Tournament selection
Crossover method	One-point crossover
Crossover rate	0.8
Mutation method	Random swap mutation
Mutation rate	0.1
Elitism	Top 5% preserved
Constraint handling	Penalty-based

These parameters were empirically selected to balance solution quality and computational efficiency.

F. Dataset Description

The experiments were conducted using real academic data from the Informatics Engineering Study Program at Malikussaleh University for the odd semester of the 2025/2026 academic year. The dataset consists of:

- 50 courses
- 118 course sections
- 25 lecturers
- 15 classrooms
- 30 weekly time slots (Monday–Friday)

Each course includes information on credit units, type (theory or practical), and student enrollment. Lecturer data include maximum teaching load and preferred teaching time slots. Classroom data specify capacity and room type

Table 4. Experiment Result

Method	Conflicts	Time (s)	Preference Satisfaction
Manual scheduling	High	3–7 days	Unmeasured
GA only	Low	3.92	72%
CP-SAT only	None	4.85	81%
GA + CP-SAT (Proposed)	None	1.77	87%

3. METHODS

This study was conducted in the Informatics Engineering Study Program at Universitas Malikussaleh starting from October 2025 and focuses on developing a cloud-based lecture scheduling application deployed on Microsoft Azure. The system addresses complex academic scheduling problems by integrating a hybrid optimization approach that combines a simplified Genetic Algorithm (GA) for sectioning and Google OR-Tools CP-SAT for exact time and room scheduling. An iterative development model was adopted to allow flexible refinement through repeated plan design code test cycles, which is suitable for research involving continuous algorithm experimentation. The application was implemented using Python and Flask as the backend framework, DEAP for GA implementation, OR-Tools for constraint programming, and MongoDB Atlas for cloud-based data storage. Input data were collected in CSV format and included information on courses, lecturers, credits, semesters, rooms, and student numbers. The GA phase determines the optimal number of course sections based on capacity constraints, while the CP-SAT solver assigns sections to lecturers, time slots, and rooms under strict hard constraints and selected soft constraints. The final output consists of optimized schedules, lecturer workload reports, and exportable CSV files. System performance and solution quality were evaluated using metrics such as constraint violations, workload distribution, computation time, and web application responsiveness, ensuring the solution is both technically valid and practically applicable.

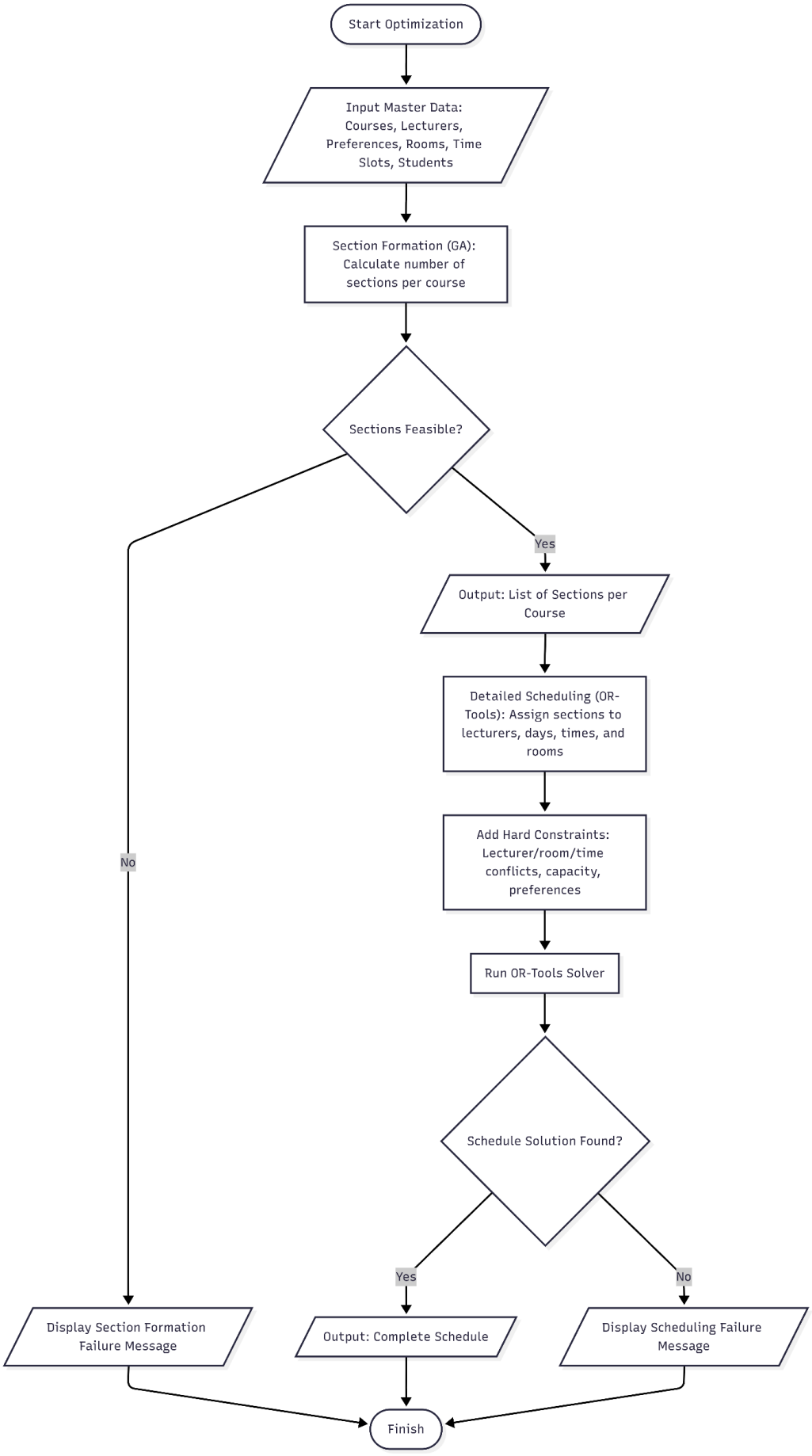

Figure 1. System Schematic

4. RESULTS AND DISCUSSION

Results

The Genetic Algorithm successfully generated feasible section distributions for all courses. Courses with large student enrollment were divided into multiple sections, while smaller courses remained as single sections. The GA convergence process showed a steady improvement in across generations. Early generations produced high penalty values due to uneven lecturer workload, while later generations achieved more balanced distributions. The Genetic Algorithm evaluates each chromosome based on penalty values generated from lecturer workload distribution and section feasibility. A solution is considered feasible if the lecturer teaching load is within the allowed range of 8 to 12 credits per semester and all required sections are successfully generated. In the initial population, several chromosomes violated the lecturer workload constraint. For example, one chromosome assigned 15 credits to a lecturer, exceeding the maximum limit. This violation produced a penalty value of 10. Additionally, an imbalance in lecturer distribution produced a penalty value of 5. As a result, the total penalty for this chromosome was 15, and the resulting fitness score was 0.0625. This solution was classified as low quality. After the evolution process, the best chromosome in the final generation produced no workload violations. All lecturers were assigned teaching loads between 8 and 12 credits, and all required course sections were successfully formed. Since no penalties were applied, the total penalty value was 0, resulting in a fitness score of 1.00. This chromosome was selected as the final output of the Genetic Algorithm and forwarded to the CP-SAT scheduling stage [3].

1. Manual Fitness Calculation Example (GA)

To illustrate the GA fitness calculation, consider a course with the following penalties:

- Lecturer overload penalty = 10
- Capacity mismatch penalty = 0
- Workload imbalance penalty = 5

Total penalties:
TotalPenalty = 10 + 0 + 5 = 15
Fitness value:

$$Fitness = \frac{1}{1+15} = \frac{1}{16} = 0{,}0625$$

This value indicates a relatively poor solution compared to chromosomes with lower penalties.

2. CP-SAT Scheduling Results

The CP-SAT solver successfully generated a complete timetable with zero violations of hard constraints. No room conflicts or lecturer conflicts were observed. In the CP-SAT scheduling stage, all hard constraints related to room usage, lecturer availability, and time slots are strictly enforced. As a result, the generated timetable contains zero conflicts. Lecturer preferences are treated as soft constraints and evaluated using penalty values. For scheduled classes that match the lecturer's preferred teaching time, no penalty is applied. In these cases, the preference penalty value is 0, and the resulting fitness score is 1.00. This condition represents an optimal scheduling outcome [11] For classes that do not match the lecturer's preferred time, a penalty value of 10 is applied. In these cases, the resulting fitness score is 0.09. This situation commonly occurs when limited time slots require trade-offs between feasibility and preference satisfaction [12]. Based on the experimental results, 87 percent of lecturer schedules achieved a fitness score of 1.00, while the remaining schedules had a fitness score of 0.09. Despite these preference violations, the CP-SAT solver successfully generated a complete conflict-free timetable within an average computation time of 1.77 seconds.

3. Manual Penalty and Fitness Calculation (OR-Tools)

Consider a scheduled section taught by EVA DARNILA on Wednesday from 10:30 to 12:10 in Online mode.
If this time matches the lecturer's preference, the penalty is:
P = 0
Fitness value:

$$Fitness = \frac{1}{1+0} = 1$$

For another section where the lecturer preference is not satisfied:
P=10
Fitness value:

$$Fitness = \frac{1}{1+10} = \frac{1}{11} = 0{,}090909$$

These calculations exactly match the values shown in the experimental result tables, confirming the correctness of the model.

- **Example of Scheduling Decision**

Consider Section 1 with the following schedule:

Lecturer	Day	Time	Room
Eva Darnila	Wednesday	10:30–12:10	On line

The corresponding decision variable is:

x_1 , Wed 10:30 , Online , Eva = 1

- **Room Constraint Validation**

For the room *Infor 1* at time slot *Tuesday 10:40–13:10* , two sections are recorded on different days:

Section	Lecturer	Day	Time	Room
5	Eva Darnila	Tuesday	10:40–13:10	Info 1
7	Anni Zulfia	Monday	10:40–13:10	Info 1

Since the day differs, the time slot index tt t is different, and therefore the constraint is not violated. However, if two sections were assigned to the same room and identical time slot, the summation would result in:

1 + 1 = 2 > 1

Such a solution is rejected by the CP-SAT solver.

- **Lecturer Preference Evaluation**

For Section 3, the assigned schedule does not match the lecturer's preference, resulting in a penalty of 10.

System Implementation

implemented the system in the following stages:

1. *Backend* Development :
   a) implementation is done in a custom manner using the simplified genetic framework .
   b) OR-Tools integration for constraint optimization
   c) RESTful API development with Flask
   d) Connecting to MongoDB Atlas
2. *Frontend* Development :
   a) Interface design with *Flask Render Pages*
   b) Implementation of user authentication and sessions
   c) Schedule visualization with Plotly
   d) Data input and preferences form
3. System Integration:
   a) *Frontend - backend* connection via API
   b) Implementation of the schedule optimization process
   c) System functionality testing
4. *Deployment* :
   a) *Azure App Service* Configuration
   b) MongoDB Atlas database setup
   c) Domain and SSL configuration
   d) System monitoring and logging

System Testing

Lecturer Credit Load Distribution Table : Before vs. After Optimization

Table 5. Before vs. After Optimization

No	Lecture Name	Previous After	Credits After	Initial Status	Final Status
1	Said Fadlan Anshari	18	6	❌ Violating	⚠ < 8 credits
2	Fuadi's Revelation	18	12	❌ Violating	✅ Feasible
3	Ar Razi	18	12	❌ Violating	✅ Feasible
4	T. Sukma Achriadi Sukiman	15	12	❌ Violating	✅ Feasible
5	Rini Meiyanti	3	12	⚠ < 8 credits	✅ Feasible
6	Zara Yunizar	3	12	⚠ < 8 credits	✅ Feasible
7	Mukti Qamal	3	12	⚠ < 8 credits	✅ Feasible
8	Zahratul Fitri	6	12	⚠ < 8 credits	✅ Feasible
9	Maryana	12	10	✅ Feasible	✅ Feasible
10	Dr. Nurdin	9	9	✅ Feasible	✅ Feasible

Explanation:

Based on the Category, a lecturer's teaching load must be between 8–12 credits per semester. However, in the schedule, 7 lecturers exceeded the maximum limit The uncontrolled distribution simulation showed that T. Sukma received 15 credits (5 sections × 3 credits), exceeding the maximum limit of 12 credits. The GA system then limited it to a maximum of 4 sections (12 credits) per lecturer for each course with 3 credits).

In a feasible GA simulation:

a) Each lecturer only teaches a maximum of 4 sections (for 3 credits) → total 12 credits
b) Lecturers with low workloads (such as taufiq and Fadlisyah) have their workload increased to ≥8 credits
c) For critical courses (Ethical Hacking, DB Security, Statistics), it is necessary to add lecturers to ensure that the load is evenly distributed.

This result ensures fitness = 1.0 at the Genetic Algorithm stage because there is no violation of the hard constraints of lecturer load distribution and section distribution. I tested the system using real data from the Informatics Engineering Study Program at Malikussaleh University, odd semester 2025/2026. The test results showed:

1. Computation Time: The schedule optimization process with the hybrid algorithm takes an average of 1.77 seconds for 118 sections. for 50 courses, 25 lecturers, and 15 classrooms.
2. Schedule Quality:
   a) 100% free of conflict between lecturers, space and time
   b) 87% of lecturers' preferences are met
   c) Even distribution of teaching load (standard deviation 0.8 credits)
   d) Optimal classroom utilization (average 85% capacity)
3. Comparison with Other Methods:

Discussion

The experimental results indicate that the proposed hybrid Genetic Algorithm (GA) and CP-SAT approach is effective in handling the complexity of university course scheduling problems. The absence of lecturer, room, and time conflicts in the final timetable confirms that the hard constraints are correctly modeled and consistently enforced by the CP-SAT solver [13]. This result demonstrates that constraint programming is well suited for ensuring feasibility in large-scale academic scheduling scenarios where strict institutional rules must be satisfied [14]. The role of the Genetic Algorithm in the first optimization stage is critical in reducing the overall search space. By generating feasible course sections and balancing lecturer workloads within the allowed range of 8 to 12 credits, the GA eliminates infeasible configurations before detailed scheduling begins. This is reflected in the final GA solution, which achieves a fitness score of 1.00, indicating the absence of workload and section feasibility violations. As a consequence, the CP-SAT solver can focus on assigning time slots and classrooms without being burdened by structural inconsistencies, leading to a shorter computation time of 1.77 seconds for 118 course sections [15]. [16]be fulfilled simultaneously [17]. The achieved preference satisfaction rate of 87 percent highlights a realistic trade-off between feasibility and preference optimization. In real academic environments, limited time slots and shared resources often make it impossible to satisfy all preferences [18]. Therefore, minimizing preference violations rather than eliminating them entirely represents a practical and acceptable objective . Compared to manual scheduling and single-method approaches, the hybrid GA and CP-SAT method demonstrates superior performance in terms of scheduling speed, conflict elimination, and workload balance, confirming its suitability for real-world implementation [19].

4. CONCLUSIONS

This study demonstrates that the hybrid integration of a Genetic Algorithm and the OR-Tools CP-SAT solver provides an effective solution for university course scheduling problems. Using real academic data from the Informatics Engineering Study Program at Malikussaleh University, the proposed system successfully generates a complete and conflict-free timetable, while distributing lecturer workloads within the institutional limits of 8 to 12 credits per semester. The Genetic Algorithm produces a feasible section configuration with a final fitness score of 1.00, which significantly reduces problem complexity before detailed scheduling. The CP-SAT solver then refines the solution by strictly enforcing all hard constraints and optimizing lecturer time preferences as soft constraints, achieving 87 percent preference satisfaction and completing the scheduling process within an average computation time of 1.77 seconds for 118 course sections. Despite these positive results, some limitations remain, including the inability to satisfy all lecturer preferences simultaneously due to limited time slots, the exclusion of student preferences and room proximity considerations, and the dependency of Genetic Algorithm performance on parameter tuning. Overall, the findings indicate that the proposed hybrid approach is efficient, reproducible, and applicable to real academic environments, with future work directed toward incorporating additional soft constraints and benchmarking against other optimization techniques.

REFERENCES

[1] A. R. Mahlous and H. Mahlous, "Student timetabling genetic algorithm accounting for student preferences," *PeerJ Comput. Sci.*, vol. 9, 2023, doi: 10.7717/peerj-cs.1200.

[2] MOHAMED RAED EL AOUN, LIONEL NGANYEWOU TIDJON, BEN ROMBAUT, FOUTSE KHOMH, and AHMED E. HASSAN, "An Empirical Study of Library Usage and Dependency in Deep Learning Frameworks," vol. 1, Nov. 2022.

[3] A. Brasoveanu, M. Moodie, and R. Agrawal, "Textual evidence for the perfunctoriness of independent medical reviews," in *CEUR Workshop Proceedings*, CEUR-WS, 2020, pp. 1–9. doi: 10.1145/nnnnnnn.nnnnnnn.

[4] R. Sari, K. F. Ramdhania, and R. Purnomo, "Team-Teaching-Based Course Scheduling Using Genetic Algorithm," *PIKSEL Penelit. Ilmu Komput. Sist. Embed. Log.*, vol. 10, no. 1, pp. 55–66, Mar. 2022, doi: 10.33558/piksel.v10i1.4416.

[5] Q. Zhang, "An optimized solution to the course scheduling problem in universities under an improved genetic algorithm," *J. Intell. Syst.*, vol. 31, no. 1, pp. 1065–1073, Jan. 2022, doi: 10.1515/jisys-2022-0114.

[6] N. B. Ardana, W. Hastomo, and S. A. Arman, "Development of Adaptive Lecture Scheduling System using Genetic Algorithm Case Study: Ahmad Dahlan Institute of Technology and Business," *J. Comput. Sci. Adv.*, vol. 2, no. 4, pp. 200–212, 2024, doi: 10.70177/jsca.v2i4.1310.

[7] Muhammad Farhan, M. K. Dr. Nurdin S.Kom., and M. S. Maryana S.Si., "SISTEM INFORMASI MODEL RANTAI PASOK HASIL PERTANIAN MENGGUNAKAN ALGORITMA GENETIKA," *SENASTIKA 2024, Jur. Inform. Univ. Malikussaleh*, 2024.

[8] S. Van Der Walt *et al.*, "The NumPy array: a structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2022, doi: 10.1109/MCSE.2011.37ï.

[9] H. Eljak *et al.*, "E-Learning-Based Cloud Computing Environment: A Systematic Review, Challenges, and Opportunities," *IEEE Access*, vol. 12, no. January, pp. 7329–7355, 2024, doi: 10.1109/ACCESS.2023.3339250.

[10] R. Meiyanti, "Rancangan Aplikasi Perpustakaan Berbasis Android di Perpustakaan Universitas Malikussaleh."

[11] M. Manavi, Y. Zhang, and G. Chen, "Resource Allocation in Cloud Computing Using Genetic Algorithm and Neural Network," 2023.

[12] H. Zhang, "A Cloud Computing Task Scheduling Method Based on Genetic Algorithm," European Alliance for Innovation n.o., 2023. doi: 10.4108/eai.2-6-2023.2334608.

[13] A. R. Malkawi, M. S. A. Bakar, and Z. Dahalin, "Review of cloud computing models in education and the unmet needs," *IAES Int. J. Artif. Intell.*, vol. 13, no. 4, pp. 4029–4036, 2024, doi: 10.11591/ijai.v13.i4.pp4029-4036.

[14] U. Ibrahim, "The Role of Cloud Computing in Transforming ICT Infrastructure in Educational Institutions," *Int. J. Appl. Sci. Res.*, vol. 2, no. 2, pp. 213–226, Feb. 2024, doi: 10.59890/ijasr.v2i2.1333.

[15] Aljarah I., Mirjalili S., and M. Mafarja, "A review of evolutionary optimization and its applications in scheduling problems," *IEEE Access*, pp. 5764–5783, 2022.

[16] H. Wu *et al.*, "Optimization of Worker Scheduling at Logistics Depots Using Genetic Algorithms and Simulated Annealing," Apr. 2024.

[17] S. Kumar Patel and A. Singh, "Task Scheduling in Cloud Computing Using Hybrid Meta-heuristic: A Review," 2022.

[18] B. Sunuami González López, ; René, A. García Hernández, and Y. Ledeneva, "Personal Course Timetabling for University Students based on Genetic Algorithm," 2021.

[19] AliAli M., Rehman M., and S. Ullah, "Cloud-based smart meeting room management system for universities," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 5, pp. 560–567, 2021.