DOI: 10.14421/ijid.2025.5067 IJID (International Journal on Informatics for Development), *e-ISSN: 2549-7448*
Vol. 14, No. 1, June 2025, Pp. 575-586

Towards Fair and Efficient Timetabling: A Genetic Algorithm Model Integrating Lecturer Day-Off Requests

Khaeroni
Department of Mathematics
Faculty of Sains, UIN SMH Banten
Serang, Indonesia
khaeroni@uinbanten.ac.id

Birru Muqdamien
Department of Informatics Engineering
Faculty of Sains, UIN SMH Banten
Serang, Indonesia
birru.muqdamien@uinbanten.ac.id

Ajeng Hestiningtyas
Department of PGMI
Faculty of Education and Teacher Training, UIN SMH Banten
Serang, Indonesia
ajenghst@gmail.com

Article History
March 10th, 2025
May 6th, 2025
May 15th, 2025
Published June, 2025

***Abstract*— This study tackles the complex challenge of lecture timetabling by incorporating lecturer day-off preferences, a crucial constraint often neglected in traditional scheduling methods. Given the NP-hard nature of the problem and the need for scalable solutions, a Genetic Algorithm (GA) was employed with a population size of 10, a crossover probability of 0.70, a mutation probability of 0.20, and a maximum generation of 10000. The proposed GA-based method, implemented using PHP and MySQL, is applied to a real-world scenario involving 25 courses, 22 lecturers, and six classrooms over a 5-day weekly schedule at the Faculty of Education and Teacher Training for the Even Semester of the 2023/2024 Academic Year. Experimental results, validated through the Mann-Whitney test, show that incorporating lecturer preferences enhances scheduling flexibility without significantly increasing computational time. Comparative analysis with Simulated Annealing and Tabu Search demonstrates the competitive performance of the GA-based method in optimizing lecture schedules. This study provides a practical solution for educational institutions seeking to improve their timetabling processes.**

Keywords*— *lecturer day-off preferences; lecture timetabling; Mann-Whitney test; scalable solutions; traditional scheduling methods

1 INTRODUCTION

Lecture timetabling constitutes the intricate task of orchestrating the placement of predetermined courses, instructed by designated lecturers, within appropriate educational facilities and infrastructure at specified times to facilitate student attendance [1]. This procedural complexity necessitates a meticulous and precise approach [2], as evidenced by the repercussions of suboptimal scheduling outcomes. Ineffectual timetables manifest as protracted and inefficient timetables, misallocating lecture rooms, resulting in suboptimal time utilization, and an undue burden on lecturers' teaching commitments. Such inefficiencies, in turn, lead to adverse indicators that impact resource utilization, thereby escalating operational costs for the academic institution [3]–[5], encompassing expenses related to electricity consumption, maintenance, and facility rentals. This challenge is further exacerbated by dynamic factors, including alterations in lecturers' course assignments and requests for schedule modifications based on individual preferences, adding a layer of complexity to the scheduling paradigm.

In many universities, traditional manual scheduling methods often result in suboptimal solutions that lead to scheduling conflicts and inefficient resource allocation [6]. The preparatory phase of schedule development encompasses the comprehensive collection of pertinent data and associated constraints. The requisite data include courses, credit weights, room specifications and capacities, available time slots, designated lecturers, and students enrolled in specific classes. Concomitantly, a range of constraints ensures adherence to stipulated scheduling parameters. Noteworthy constraints involve the prohibition of a lecturer concurrently instructing in multiple rooms, restricting students from attending lectures in more than one room simultaneously, scheduling limitations for particular lecturers based on their preferences, and similar considerations. The scheduling process begins once the requisite data and constraints have been compiled, ensuring that the provided data and constraints align with activities. This meticulous procedure ensures the judicious allocation of resources and facilitates the construction of an efficacious timetable.

Lecture timetabling is categorized within the broader domain of timetabling [7], a class of problems recognized for its NP-Hard classification (nondeterministic polynomial time). Courses must be scheduled optimally while adhering to multiple constraints, including the availability of lecture rooms, the preferences of instructors, and the alignment with student timetables [8]. Petrovic and Burke state that lecture timetabling is a complex NP-hard problem that requires an efficient scheduling mechanism to accommodate multiple constraints, including course assignments, room availability, and lecturer preferences [9]. The NP-Hard classification denotes problems that, when attempted to be solved through exhaustive testing of all possible combinations, exhibit an exponential increase in the time required to identify a viable solution [10]. Consequently, conventional optimization methodologies face significant challenges in effectively addressing optimization problems of this nature [11]. The inherent complexity of the timetabling problem, compounded by its NP-hard status, underscores the need for advanced heuristic approaches, such as genetic algorithms, to navigate the intricate solution space and derive efficient scheduling outcomes.

Several metaheuristic optimization techniques have been widely explored for automated timetable generation, including Simulated Annealing (SA), Tabu Search (TS), and Particle Swarm Optimization (PSO) [12]–[15]. While these methods offer practical solutions, they often struggle with scalability and adaptability to dynamic scheduling constraints [16].

In addressing challenges of this nature, Alghamdi posits that, to date, there lacks an algorithm capable of exhaustively testing all conceivable possibilities to ascertain the optimal solution within a reasonable timeframe [17]. Consequently, the prevailing approach to tackling such intricate problems involves employing heuristic methodologies, with a recent emphasis on meta-heuristics. Notably, Alghamdi underscores the prevalence of Genetic Algorithm (GA) as a leading methodology in this domain. GA is favoured due to its capacity for substantial computational parallelization, thereby enhancing overall computing performance. Furthermore, Alghamdi contends that GA stands out as particularly adept at addressing NP-complete problems, aligning with observations from Diveev [18], Abdelhalim [19], and Diveev [20]. GA has gained popularity for its ability to handle complex constraint-based optimization problems while maintaining solution diversity and robustness [21]–[23]. This consensus among scholars underscores the efficacy of GA in providing pragmatic solutions to complex optimization challenges, reinforcing its status as a forefront methodology in contemporary problem-solving paradigms.

GA has been extensively applied in academic timetabling due to its ability to search large solution spaces efficiently while maintaining solution diversity [14], as exemplified by endeavours conducted by Puspasari [24], Paranduk [25], Khairunisak [26], Qashlim [27], Hidayati [28], Nugraha [29], Afandi [30], Ardiansyah [31], Suwirmayanti [32], Mone [33], and Puspaningrum [34]. GA operates by evolving a population of potential timetables over multiple generations through selection, crossover, and mutation operations, progressively improving the quality of solutions [35]. Although GA is widely used in timetabling optimization, other metaheuristic techniques, such as Simulated Annealing (SA) and Tabu Search (TS), have also been employed for solving scheduling problems [14]. Table 1 presents a comparative analysis of these techniques regarding search strategy, adaptability, and solution quality.

Table 1. Comparative Analysis: GA vs Simulated Annealing vs Tabu Search vs PSO vs ACO

Algorithm and References	Advantages	Limitations	Everyday Use Cases in Scheduling
Genetic Algorithm (GA) [14]–[16], [21]–[23], [35]	- Handles large solution spaces well - Adaptable to dynamic constraints	- Requires careful parameter tuning - May converge prematurely if not tuned properly	- University course timetabling - Examination timetabling - Resource allocation

	- Maintains population diversity - Good for multi-objective optimization		
Simulated Annealing (SA) [12], [13], [17]	- Efficient at escaping local optima - Simple to implement - Effective for small to medium-sized problems	- Computationally expensive for large-scale problems - Slow convergence rate	- Small-scale scheduling - Optimization with a few variables
Tabu Search (TS) [12], [14], [18]	- Avoids cycles and revisiting solutions - Fast convergence on structured problems - Memory-based search	- Struggles with maintaining diversity in large search spaces - Complex neighborhood design required	- Nurse rostering - Job shop scheduling - Vehicle routing
Particle Swarm Optimization (PSO) [13], [16], [35]	- Easy to implement - Fast convergence - Suitable for continuous and combinatorial problems	- Prone to premature convergence - Parameter sensitivity affects performance	- Real-time scheduling - Dynamic job scheduling - Parallel processing
Ant Colony Optimization (ACO) [16], [19], [36]	- Strong path-finding capability - Distributed computation - Adaptive to dynamic environments	- Slower convergence than other methods - High memory and computational requirements	- Routing problems - Constraint-based scheduling - Sequential task planning

Studies indicate that while SA and TS are effective for small to medium-sized timetabling problems, GA performs better in handling dynamic constraints, making it preferable for large-scale university scheduling [15]. Several studies have highlighted that the integration of multiple methods (hybrid approaches), such as Genetic Algorithm combined with Simulated Annealing (GA + SA) or Ant Colony Optimization (GA + ACO), can significantly enhance both the efficiency and quality of the resulting solutions [12], [16], [36]. Despite these advancements, GA remains a preferred choice due to its flexibility in accommodating various constraints, including lecturer day-off preferences [26], [29]. The analysis presented in Table 1 demonstrates that the GA performs better in managing dynamic constraints while concurrently preserving computational efficiency.

Studies have demonstrated that GA-based timetabling can handle hard constraints (such as room and time slot availability) and soft constraints (such as lecturer preferences and student convenience) more effectively than rule-based approaches [36]. Recent advancements in GA applications for timetabling have focused on hybrid models, combining GA with other optimization techniques to improve performance. For example, Weng et al. proposed a hybrid GA approach that integrates local search heuristics, demonstrating significant improvements in scheduling efficiency [37].

Studies suggest fine-tuning GA parameters, such as mutation rate and selection strategy, can enhance scheduling outcomes [35]. A set of rigid constraints has been consistently employed in these scholarly pursuits. These constraints encompass the stipulations that: 1) Lecturers are precluded from instructing more than one class concurrently on the same day; 2) Each course necessitates at least one assigned lecturer; 3) A singular lecture room is not concurrently utilized by more than one class at the same scheduled time; 4) A given class is not scheduled for multiple courses simultaneously on the same day; and 5) Courses with a credit weight of 4 are conducted in two separate sessions. The researchers assert the successful development of a scheduling application utilizing genetic algorithms adept at accommodating these constraints. Evaluation of the application's efficacy, based on the outcomes of rigorous testing, underscores its capability to yield optimal schedules characterized by diverse computational times. The merit of a schedule is gauged by the resultant optimal function values, serving as a benchmark for determining scheduling efficacy.

An essential yet frequently overlooked aspect of academic scheduling is accommodating lecturer day-off preferences, which can significantly impact faculty workload management and overall job satisfaction. Despite the effectiveness of existing timetabling algorithms, most studies do not explicitly accommodate lecturer day-off preferences, a crucial factor for faculty workload management and overall satisfaction [38]. Research has demonstrated that workload and work-life balance influence lecturers' job satisfaction and performance. For instance, a study by Zaidan and Juariyah found that lecturer workload is a burden assigned by higher education leadership, and excessive workload can lead to work stress, negatively affecting job satisfaction [39]. Similarly, research by Siahaan et al. highlighted that work-life balance significantly affects job satisfaction among lecturers, emphasizing the importance of balancing work and personal life to achieve optimal job satisfaction [40].

Moreover, hybrid GA-based timetabling models that integrate machine learning or reinforcement learning remain underexplored [36]. Therefore, this study aims to address these limitations by developing a GA-based lecture timetabling model that explicitly incorporates lecturer day-off constraints, providing a scalable and adaptable solution for academic scheduling. Therefore, integrating lecturer day-off preferences into scheduling processes is crucial for effective workload management and overall job satisfaction [41].

Concerning the impediment posed by lecturers' day off, several studies have addressed the concept of "lecturer day off" either as a soft or a hard constraint in academic scheduling. Khairunisak and Nugraha [26], [29] classified lecturer day-off as a hard constraint, arguing that violations may lead to job dissatisfaction and disrupt the work-life balance. In contrast, Qashlim [27] categorized it as a soft constraint, as violations are only mildly penalized and do not necessarily halt the scheduling process. The nuanced treatment of this constraint in the literature underscores the need for further research elucidating its integration's intricacies and quantifiable impact on scheduling optimization. However, most existing GA-based timetabling studies do not explicitly integrate lecturer day-off constraints,

creating a practical implementation gap in academic scheduling systems.

In the present study, lecturer day-off is treated as a hard constraint, as the authors aim to ensure that lecturers' time-off preferences are fully accommodated. This approach is intended to enhance job satisfaction and optimize workload management. This decision aligns with findings by Zaidan and Juariyah [39] and Siahaan et al. [42], who emphasized the importance of personal time management in supporting both performance and lecturer satisfaction.

Soliciting a lecturer's day off constitutes a salient and consequential consideration, warranting meticulous attention to mitigate the necessity for subsequent revisions to the finalized schedule, thereby averting prolonged scheduling processes. However, the nuanced influence of incorporating these constraints on the overall execution time remains an unresolved aspect. Consequently, the present research investigates this matter empirically through a structured experimental approach, wherein lecturer holidays are systematically introduced as rigid constraints within the optimization framework. The objective is to discern the quantitative impact of this constraint addition on the temporal efficiency of the scheduling process, thus contributing valuable insights to optimizing lecture schedules under the specific consideration of lecturers' day off preferences.

Hard constraints are mandatory conditions that must not be violated—for example, a lecturer cannot teach two classes simultaneously. In contrast, soft constraints represent desirable preferences that are ideally satisfied but may be relaxed if necessary, such as room or time slot preferences. A trade-off arises when prioritizing soft constraints, which makes it more challenging to satisfy hard constraints or vice versa. In this study, the lecturer's day off is classified as a hard constraint to ensure that all lecturers' preferred days off are fully honoured without exception.

Although hybrid approaches, such as integrating GA with local search techniques, machine learning, or reinforcement learning, have demonstrated potential in enhancing scheduling performance, the present study focuses on developing a fundamental GA model capable of effectively and efficiently handling the lecturer day-off constraint. The primary objective is to demonstrate that a standard GA framework can yield optimal results without introducing additional complexity into the system design. Implementing hybrid approaches is a future direction, as the Future Work section outlines. This study aims to establish that a basic GA model is sufficiently robust to address complex constraints, such as lecturer day-off, while maintaining computational efficiency.

Thus, this study aims to develop a GA-based lecture timetabling approach that accommodates lecturer day-off constraints while maintaining computational efficiency. The main contributions of this research are: 1) Developing an optimized GA-based timetabling model that integrates lecturer preferences; 2) Implementing and evaluating the model using PHP and MySQL, making it adaptable for real-world applications; 3) Comparing the proposed method with other metaheuristics to assess performance and computational efficiency; and 3) Providing a scalable scheduling solution for universities with dynamic constraints and large datasets.

The factors of lecturer satisfaction and workload management in this study are grounded in the analysis of historical data and direct observation of prior scheduling patterns. Although a formal survey of lecturers was not conducted in this research, the validity of these assumptions is supported by several previous studies, including those by Zaidan & Juariyah [39] and Siahaan et al. [42], which affirm that scheduling flexibility and the availability of designated days off significantly contribute to lecturers' job satisfaction.

2 METHOD

The lecture timetabling problem is a combinatorial optimization problem where courses, instructors, and classrooms must be scheduled into available time slots while satisfying various constraints [8], [9]. This study focuses on accommodating lecturer day-off preferences, a constraint often neglected in previous timetabling models [15]. The penalty weights assigned to each constraint are presented as follows:

- No two courses are assigned to the same lecturer or classroom simultaneously.
- No course was assigned during the noon and Friday prayers.
- Courses must fit within the university's available working hours.
- Lecturer day-off requests must be respected.

The experimental apparatus employed for the investigation consists of a laptop that houses a pre-developed scheduling application incorporating a genetic algorithm. The computational device utilized possesses specific technical specifications, detailed comprehensively in Table 2. This instrumentation configuration ensures standardized conditions for executing the genetic algorithm-based scheduling application, facilitating a controlled and reproducible experimental environment.

The dataset was obtained from the PGMI Department, Faculty of Education and Teacher Training, State Islamic University of Sultan Maulana Hasanuddin Banten, comprising 25 courses, 22 lecturers, and six classrooms, operating on a 5-day weekly schedule, including Fridays. The dataset includes lecturer preferences, course durations, room availability, and student enrollments.

Table 2. Experimental Instrument Specifications

No	Type	Specification	
1	Hardware	Platform: Laptop	
		Brand	: Lenovo
		Type	: Yoga Slim 7i Carbon
		Display	: 13.3'
		Proc	: Intel i7-1165G7
		RAM	: 16 GB DDR3
		Storage	: SSD NVMe 1 TB
2	Software	OS	: Win. 11 Home SL
		Code Editor	: Visual Studio Code
		PHP version	: 8.0.7
		Database	: MySQL Server 10.4
		Framework	: CodeIgniter 3
		Browser	: Firefox 119.01 (64-bit)

To evaluate the quality of the solutions more holistically, this study employs several performance metrics: 1) Feasibility Rate, the percentage of individuals (solutions) that satisfy all hard constraints; 2) Fitness Score, the average fitness value of the best-performing generation; and 3) Lecturer Day-off Satisfaction Rate, the percentage of lecturers whose assigned schedules do not conflict with their preferred days off. The diagram in Fig. 1 below illustrates the stages involved in implementing the GA.

The testing protocol is systematically conducted sequentially. Initially, the execution time test is undertaken without including lecturer day-off data, establishing a condition wherein the lecturer day-off table remains unpopulated. Subsequently, the execution time testing is iterated with the prior incorporation of lecturer day-off data, thereby altering the condition of the lecturer day-off table to a non-empty state. The test dataset comprises 12 records, symbolizing the hours constituting a single effective lecture day. This study's test parameters or variables include population size, crossover probability, mutation probability, and the number of generations. Both tests utilize identical GA parameters delineated in Table 3 and are conducted on uniform equipment.

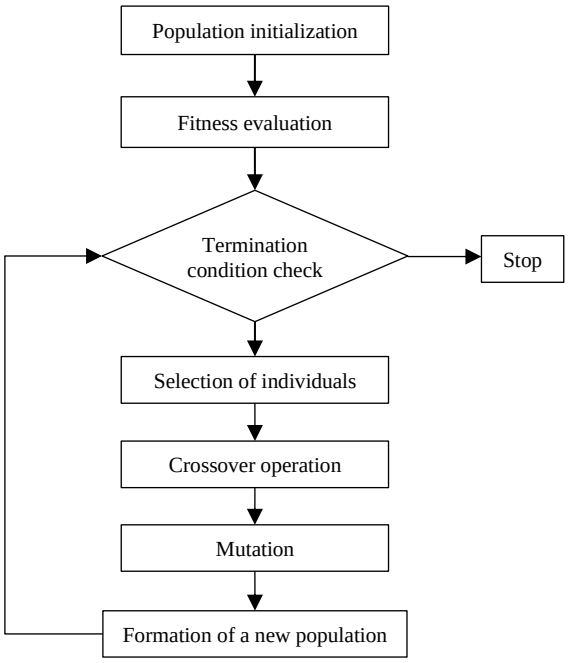

Figure 1. Flowchart of GA implementation.

Table 3. The GA Test Parameters

No	Parameter	Value
1	Total Population	10
2	Crossover Probability	0.70
3	Mutation Probability	0.20
4	Number of Generations	10000
5	maximum_execution_time	7200

The selection of probability parameters is grounded in prior simulation outcomes, wherein these specific parameters exhibited notable improvements in execution times. Concurrently, the parameter for maximum execution time is set at 7200 seconds. This deliberate configuration circumvents premature termination of the search process by the PHP compiler, which conventionally enforces a default time limit of 120 seconds. Consequently, the PHP compiler perseveres with the execution of the GA process until one of the ensuing conditions is met:

- Attainment of individuals with a fitness value of 1, indicating the optimal solution.
- Generation counts surpassing 10000 generations.
- Elapsed time exceeding 7200 seconds.
- Additional technical conditions that prompt the PHP compiler to halt execution.

The quest for scheduling solutions, with and without accommodating lecturers' day off, entails a systematic process wherein the corresponding day off data is incorporated or omitted. This iterative experimentation is conducted through 30 repetitions for each search procedure, establishing a robust empirical foundation. The sample data from these trials comprises scheduled search execution times, reflecting a continuous dataset. The investigation adheres to a methodologically stringent framework by adopting a quantitative research approach with an experimental design featuring two independent samples. This analytical methodology is selected to discern potential differences in schedule search execution times, thereby elucidating the influence of accommodating or disregarding lecturers' days off on the temporal efficiency of the scheduling process.

Following Sedgwick's recommendations, the ensuing data analysis is executed through non-parametric means, specifically employing the Mann-Whitney U test [43], facilitated by the Statistical Package for the Social Sciences (SPSS) version 22 application. This analytical approach leverages the Mann-Whitney U, a non-parametric statistical test, to assess the significance of differences between two independent groups. The integration of SPSS as the analytical tool ensures a robust and rigorous examination of the data, contributing to the statistical validity and reliability of the findings.

The formulated hypotheses for this study are articulated as follows:

Ho : The accommodation of lecturers' days off does not exert a positive and statistically significant influence on the execution time of the genetic algorithm in generating the optimal schedule.

Ha : The accommodation of lecturers' days off has a positive and statistically significant influence on the execution time of the genetic algorithm in generating the optimal schedule.

3 RESULT AND DISCUSSION

3.1 The GA Terminologies

GA was selected for this study due to its proven ability to handle complex scheduling problems efficiently [44]. The implementation of GA in this study adheres to the following terminologies.

3.1.1 *Chromosome Representation:* The preliminary stage involves the representation of the lecture schedule in chromosome format. Chromosomes are composed of multiple genes, each serving as a hereditary element within the context of inheritance. The constituents of the lecture schedule chromosome, elucidated through an illustrative depiction in Figure 2, encompass various genes that encode essential scheduling parameters. This chromosome representation establishes the foundation for the subsequent application of genetic operators to iteratively evolve and optimize the lecture schedule in congruence with specified constraints and objectives.

An individual within the context of a genetic algorithm is composed of multiple chromosomes, with the number of chromosomes contingent upon the volumetric representation of pertinent data from each gene. Determining the number of chromosomes within the scheduling system is predicated on the informational richness encapsulated by each gene. An individual corresponds to a comprehensive lecture schedule package tailored for a specific semester in the lecture scheduling system. Each individual is uniquely identified through three indices delineated: $individual_{ijk}$.

Here, index i designates the i-th individual, index j signifies the j-th plotting code, and index k assumes distinct values with specific semantic implications: $k = 0$ denotes total credits, $k = 1$ corresponds to time code, $k = 2$ pertains to days, and $k = 3$ signifies room. The alphanumeric representations within the nomenclature, such as $individual_{110} = 2$, $individual_{111} = 1$, $individual112 = 2$, and $individual_{113} = 12$, elucidate the specific attributes of an individual within the context of the GA employed for lecture timetabling. In this context, the designation $individual_{110} = 2$ signifies that, for the first individual in a given generation, the first instructor is scheduled to teach on the second day during the first hour, instructing for two credits, in room 12.

This indexing schema establishes a structured framework for the comprehensive representation and differentiation of individuals within the lecture timetabling system, facilitating the efficient execution of genetic operators during optimization.

Chromosome	= (Gen1, Gen2, ..., GenN)
Chromosomes Lecture Schedule	= (Lecture, Day, Time, Room)
For Gen, lectures have sub-Gen, namely:	
Lecture	= (Lecturer, Subject, Class)

Figure 2. Chromosomal representation of the schedule.

3.1.2 *Coding Scheme:* Each datum representing a gene undergoes a specific coding procedure. The employed coding technique follows a normalization methodology, wherein the unique code applied is derived from data sourced from a pre-established database. The coding scheme outcomes for each gene within the lecture timetable chromosome are encapsulated in Table 4, providing a comprehensive representation of the normalized codes assigned to distinct genes. This meticulous coding process ensures the uniformity and systematic organization of the genetic information, facilitating its efficient utilization within the GA framework applied to college schedule optimization.

3.1.3 *Population Set:* Within a GA framework, a population comprises multiple individuals, with the number of individuals constituting a population as a pivotal determinant at this stage. In the lecture timetabling system context, the population set corresponds to the number of distinct versions of lecture schedule packages generated for a specific semester. Establishing an optimal population size is paramount, as it governs the solutions' diversity and exploration potential within the GA.

3.1.4 *Fitness Evaluation:* The primary objective of fitness evaluation within this study's context is to ascertain an individual's quality, specifically appraising the excellence of the generated version of the lecture schedule. Fitness calculations are inherently entwined with considering constraints, where only hard constraints are scrutinized. These hard constraints, constituting scheduling rules or imperatives, are non-negotiable and must not be transgressed to prevent the emergence of conflicting class schedules. The employed fitness evaluation method utilizes the minimum function. The objective function is derived from multiplying the number of violations (penalties) by the weight assigned to each violation. In this context, a penalty represents a transgression of a constraint or scheduling rule. The fitness function adopted is expressed as shown in (1):

$$\text{Fitness} = \frac{1}{\text{Penalty} + 1} \quad (0)$$

Table 4. The Coding Scheme

Gen	Coding
Lecturing	Primary key in the lecture table (lecturer, course, class)
Day	Primary key in the day table
Time	The primary key in the timetable
Room	Primary key in the room table

This formulation introduces a penalty weight, representing the violation's magnitude for each constraint. The corresponding penalty weights for individual constraints are meticulously outlined in Table 5, providing a comprehensive framework for the quantitative assessment of fitness concerning constraint violations within the lecture timetables system. Each constraint violation is assigned a penalty weight of 1. Table 6 shows the penalty weights for several soft constraints. The final fitness score thus represents the total number of infringements present in a given solution individual. A lower fitness score indicates fewer constraint violations, reflecting a higher-quality solution.

3.1.5 *Optimal Solution and Individual Solution:* The process of determining the optimal solution is achieved through an assessment of the highest fitness value within a given generation or population. Identifying an optimal solution entails discovering individual solutions, with an individual being designated as a solution individual if it attains the highest fitness value within a population, signifying an optimal fitness value (fitness = 1). Additionally, individual solutions are considered achieved when there is a persistence of the highest fitness value across successive generations, indicating a state of equilibrium where no further improvement in fitness is observed. This meticulous evaluation of fitness values is a pivotal criterion for discerning and classifying optimal solutions in the iterative progression of the GA applied to lecture timetable optimization.

3.1.6 *Individual Selection:* Individual selection constitutes the initial phase in the iterative process, involving forming a new generation or population within the framework of a GA. This process is activated if an individual solution, denoting the optimal version of the lecture schedule, has not yet been identified. Before the individual selection stage, a comprehensive computation of the fitness value for each individual is undertaken. The selection process unfolds by employing the "good fitness" method, entailing the removal of half of the individuals with the lowest fitness values from the population. Conversely, half of the individuals featuring the highest fitness values are retained as the selected individuals or parents.

Table 5. Penalty Weights Table for Hard Constraints

No	Hard Constraint	Penalty Weights
1	One lecturer teaches more than one lecture at the same time	1
2	One class attends more than one lecture at the same time.	1
3	One room is used for more than one lecture at the same time.	1
4	The lecture is held during the noon prayer.	1
5	The lecture is held during the Friday prayer.	1
6	The lecturer teaches during their day off.	1

Table 6. Penalty Weights Table for Soft Constraints

No	Soft Constraint	Penalty Weights
1	Preference for morning classes	0.5
2	Preference for afternoon classes	0.5
3	Preference for room type	0.3
4	Lecturer's day-off preference	1

3.1.7 *Crossover:* The crossover process is enacted upon the selected individuals or parents, resulting in the generation of new individuals, commonly referred to as offspring. This crossover operation aims to adhere to the prescribed population size regulations within a given generation. The adopted crossover technique in this context is a two-point crossover. This methodical approach involves the exchange of genetic material between two distinct points along the chromosomal structures of the selected individuals.

3.1.8 *Mutations:* The mutation phase involves the generation of novel individuals through the alteration of existing individual gene values. The fundamental objective of this mutational process is to introduce the prospect of acquiring new individuals characterized by elevated or improved fitness values. By manipulating the genetic information encoded within individual genes, the mutation stage introduces variability into the population, fostering diversity and allowing the GA to explore alternative solutions.

3.1.9 *New Population:* A novel population or generation is generated following an iterative sequence encompassing individual selection, crossover, and mutation. After this generational transition, a comprehensive evaluation of fitness values within the new population is undertaken. The GA process is designed to terminate upon identifying the highest fitness value within a population that attains the optimal threshold (fitness = 1). Individuals featuring these optimal fitness values are designated as the selected versions of the lecture timetable, constituting individual solutions. Conversely, in scenarios where a particular solution has not been

identified, the iterative cycle is reiterated to instantiate a new population or generation, perpetuating the algorithmic refinement process in pursuing optimal lecture schedules.

3.2 Day-off Implementation in Timetabling

The subsequent step involves the incorporation of day-offs into the scheduling application, with a specific focus on lecturer day-offs, signifying periods when lecturers are unavailable for teaching. In this context, day-offs are treated as hard constraints, representing imperative obstacles that necessitate meticulous avoidance. The attribution of a penalty value of 1 underscores the non-negotiable nature of day-offs within the timetable framework. The GA's implementation incorporates penalty assignments when a lecturer is scheduled to teach during undesired time slots, thus aligning with the notion of a day off as a stringent constraint. Technically, within the application, the execution of these procedures entails a structured sequence of steps to ensure the systematic integration of day-off constraints into the timetabling algorithm. This study is focused on evaluating the baseline implementation of the GA to determine whether lecturer day-off preferences can be integrated without compromising computational efficiency. The development of hybrid approaches is reserved for future research.

3.2.1 *Create a table to store lecturer day-off data*: Within this study, the employed table structure is delineated in Table 7. This table serves as a medium for storing lecturer day-off data, ensuring accessibility for subsequent processes within the research framework. The systematic organization of data within Table 7 facilitates the efficient retrieval and utilization of lecturer day-off information in various stages of the research workflow.

3.2.2 *Add a script to the fitness calculation function:* In the context of this research, a script has been incorporated to enforce penalties in instances where the specified hard constraint is violated. The subsequent code snippet in Figure 3 illustrates the technical implementation. The code snippet exemplifies the integration of penalty enforcement into the algorithmic framework, wherein the violation of the hard constraint, such as a lecturer's day off, triggers the application of penalties. The provided code is an illustrative reference for the technical implementation of penalty mechanisms within the research context.

3.2.3 *Test result:* After implementing the lecturer's day-off constraint and the specification of testing parameters, the subsequent phase involves conducting tests and recording the time required by the GA to yield an optimal solution. The ensuing execution time data for the initial two tests is delineated in Table 8, providing a comprehensive overview of the algorithm's temporal efficiency under varied conditions.

The derivation of proportion data emanates from juxtaposing the generated generations against the corresponding execution time. Consequently, the proportion data signifies the average time required to complete one generation within the GA process. In pursuing testing objectives, a comprehensive analysis is conducted separately for the execution time and the average time data. This dual analytical approach provides distinct insights into the temporal dynamics and efficiency of the GA process, offering a nuanced understanding of the algorithm's performance characteristics under diverse conditions.

Table 7. Lecturer Day-Off Table Structure

Field Name	Type	Length	Note
kode	int	11	Primary Key
kode_dosen	int	11	
kode_hari	int	11	
kode_jam	int	11	

```
...
$rs_WaktuDosen      =      $this->CI->db->query("SELECT
kode_dosen,  CONCAT_WS(':',  kode_hari,  kode_jam)  as
kode_hari_jam FROM day_off");
$i = 0;
foreach ($rs_WaktuDosen->result() as $data) {
  $this->idosen[$i] = $data->kode_dosen;
  $this->waktu_dosen[$i][0]= $data->kode_dosen;
  $this->waktu_dosen[$i][1] = $data->kode_hari_jam;
  $i++;
}
...
$jumlah_waktu_tidak_bersedia = count($this->idosen);
for ($j = 0;
     $j < $jumlah_waktu_tidak_bersedia; $j++) {
  if ($dosen_a == $this->idosen[$j]) {
    $hari_jam = explode(':',
      $this->waktu_dosen[$j][1]);
    if ($jam_a == $hari_jam[1]
      && $hari_a == $hari_jam[0]) {$penalty++;}
    if ($sks >= 2) {
      if ($jam_a + 1 == $hari_jam[1]
        && $hari_a == $hari_jam[0]) {$penalty++;}}
    if ($sks >= 3) {
      if ($jam_a + 2 == $hari_jam[1]
        && $hari_a == $hari_jam[0]) {$penalty++;}}
      if ($sks >= 4) {
        if ($jam_a + 3 == $hari_jam[1]
          && $hari_a == $hari_jam[0]) {$penalty++;}}
   }
}
```

Figure 3. An additional script to implement the day off.

Table 8. Execution and Average Time Data (Proportion)

n-th test	Without Day-off			With Day-off		
	Number of Generations	*Execution time (seconds)*	*Proportion*	*Number of Generations*	*Execution time (seconds)*	*Proportion*
1	806	35	23.03	1037	83	12.49
2	743	42	17.69	931	73	12.75

3	2025	105	19.29	1338	106	12.62
4	1215	77	15.78	581	46	12.63
5	1659	96	17.28	881	65	13.55
6	741	29	25.55	2185	161	13.57
7	1323	78	16.96	1129	82	13.77
8	986	55	17.93	950	72	13.19
9	761	31	24.55	427	31	13.77
10	953	74	12.88	510	51	10.00
11	1444	107	13.50	958	74	12.95
12	1616	121	13.36	1945	153	12.71
13	795	67	11.87	2145	161	13.32
14	624	48	13.00	1104	86	12.84
15	720	59	12.20	904	71	12.73
16	1375	105	13.10	1205	94	12.82
17	2222	163	13.63	2364	162	14.59
18	1146	85	13.48	1157	59	19.61
19	797	61	13.07	712	37	19.24
20	852	69	12.35	1173	59	19.88
21	1274	95	13.41	891	54	16.50
22	1997	146	13.68	800	41	19.51
23	1624	121	13.42	697	59	11.81
24	455	36	12.64	1031	74	13.93
25	961	72	13.35	1103	59	18.69
26	969	76	12.75	594	42	14.14
27	1769	117	15.12	1271	81	15.69
28	861	60	14.35	1032	67	15.40
29	1634	106	15.42	1152	81	14.22
30	607	42	14.45	1120	71	15.77

3.2.4 *The analysis of execution time:* The data analysis outcomes were facilitated by the Mann-Whitney U Test with the assistance of SPSS, as presented in Table 9.

The statistical hypothesis in this test is:

Ho : There is no difference in genetic algorithm execution time between those that accommodate the lecturer's days off and those that do not accommodate the lecturer's days off.

Ha : There is a difference in genetic algorithm execution time between those that accommodate the lecturer's days off and those that do not.

The criteria for rejecting the null hypothesis, Ho, involve assessing whether the significance value (Sig) is less than 0.05. If the Sig value is less than 0.05, Ho is rejected; otherwise, Ho is accepted. In Table 9, the Sig value is observed to be 0.690. Ho is received according to the predefined criterion since the Sig value (0.690) exceeds 0.05. Consequently, it is inferred that there is no significant difference in the execution time of the GA, whether accommodating or not accommodating lecturers' days off. In other words, both datasets exhibit comparable average values, indicating a lack of statistically significant divergence in execution time under the specified conditions.

Table 9. The Mann-Whitney U Test Results for Execution Time

Parameters	Execution Time
Mann-Whitney U	423.000
Wilcoxon W	888.000
Z	-.399
Asymp. Sig. (2-tailed)	.690

Source: SPSS Output

3.2.5 *The average time analysis:* The outcomes of the data analysis, employing the Mann-Whitney U Test with the support of SPSS, are detailed in Table 10.

The statistical hypotheses under consideration for this test are as follows:

Ho : There is no difference in the average genetic algorithm processing time for one generation between schedules that accommodate lecturers' days off and those that do not.

Ha : There is a difference in the average genetic algorithm processing time for one generation between schedules that accommodate lecturers' days off and those that do not.

The criteria for rejecting the null hypothesis, Ho, are contingent upon the significance value (Sig). If the Sig value is less than 0.05, Ho is rejected; otherwise, Ho is accepted. In Table 8, the Sig value is observed to be 0.520. Following the specified criteria, Ho is accepted as the Sig value (0.520) exceeds 0.05. Consequently, it is deduced that the average time required for the genetic algorithm process for one generation, whether accommodating or not accommodating lecturers' holidays, does not exhibit a significant difference. In essence, both datasets manifest analogous average values in this context.

3.3 Discussion

The GA-based lecture timetabling system was implemented using PHP and MySQL to optimize the scheduling of 25 courses, 22 lecturers, and six classrooms over a five-day academic period. The model effectively accommodated lecturer day-off constraints while minimizing scheduling conflicts. Its performance was evaluated based on three key metrics: (1) Constraint Satisfaction Rate (CSR), measuring adherence to hard and soft constraints; (2) Execution Time (ET), assessing the duration required to generate an optimized timetable; and (3) Solution Quality (SQ), analyzing the balance between constraint violations and scheduling efficiency.

Table 10. Mann-Whitney U Test Results for Average Time

Parameters	Execution Time
Mann-Whitney U	406.500
Wilcoxon W	871.500

Z	-.643
Asymp. Sig. (2-tailed)	.520

Source: SPSS Output

A comparative analysis of execution time and average GA processing time was conducted between two scenarios: one incorporating lecturer day-off constraints and the other without this restriction. The results demonstrated that introducing a lecturer's day off as a hard constraint did not yield significant changes in execution time. Both scenarios converged toward optimal solutions, with execution time variability remaining negligible. These findings suggest that adding a single strict constraint, such as lecturer day-off preferences, does not impose a computational burden on the scheduling process.

This conclusion aligns with Gupta et al., who found that GA-based scheduling maintains high efficiency without excessive computational overhead [15]. However, Bajeh and Abolarinwa suggest that hybrid models integrating GA with local search heuristics may further enhance scheduling performance [14]. The findings imply that incorporating Reinforcement Learning (RL) or Deep Learning techniques could refine GA-generated solutions and improve optimization outcomes.

Prior studies have established that GA-based approaches outperform traditional heuristics, particularly when managing dynamic constraints like lecturer preferences [45]. Mills further emphasized that crossover and mutation rates significantly influence GA efficiency more than introducing additional constraints [46]. Similarly, Iglesias demonstrated that a 3% mutation probability increases the likelihood of achieving an optimal solution by 10-12%, while crossbreeding probabilities exhibit minimal impact [47]. These findings reinforce that constraints such as the lecturer's day off do not negatively affect execution time.

In real-world applications, Devi et al. confirmed that GA effectively automates academic timetabling while ensuring flexibility and accuracy, consistent with the present study [8]. Additionally, Khan and Imtiaz developed a GA-based system for dynamically adjusting academic schedules based on predefined parameters, demonstrating that optimal solutions can be obtained within a short processing time [48]. These results affirm that GA remains computationally efficient, even with additional constraints.

From a practical perspective, this study highlights that integrating lecturer day-off preferences into GA-based scheduling does not compromise computational efficiency, offering significant benefits for academic institutions. Specifically, universities can:

- Minimize scheduling conflicts by efficiently incorporating lecturer availability constraints.
- Optimize resource allocation to ensure an even distribution of classrooms and instructors.
- Enhance scalability, allowing the system to adapt to larger datasets and more complex scheduling conditions.

Lopes further supports these findings, concluding that GA-based scheduling models accommodate flexible constraints without compromising execution time efficiency [49]. The robustness of GA in academic scheduling reinforces its practical advantages over conventional methods, offering a scalable and adaptable solution for dynamic scheduling environments.

Despite these promising results, some limitations persist. The computational complexity may increase with larger datasets, necessitating the exploration of parallel processing techniques to maintain efficiency [44], [50]. Additionally, hybrid models integrating GA with Machine Learning or Reinforcement Learning warrant further investigation to enhance scheduling accuracy and adaptability [15], [51]. Notably, limited research has examined the impact of introducing new constraints in academic scheduling. Also, lecturer satisfaction was validated through literature references and internal observations. A formal survey of lecturers was not carried out due to limitations in time and budget. This study provides novel insights that can be extended to broader scheduling applications, paving the way for more intelligent and adaptive timetable optimization frameworks.

4 CONCLUSION

Considering the outcomes derived from the dual testing modalities, encompassing execution time and average execution time, it is deduced that the introduction of constraints in the guise of lecturer day-off requests does not exert a statistically significant impact on either the execution time or the average execution time requisite for the identification of the optimal solution. Consequently, it is asserted that the development of a lecture scheduling application utilizing a genetic algorithm can be systematically achieved by incorporating constraints reflective of temporal preferences specified by the lecturers without significantly compromising the computational efficiency in the search for optimal scheduling solutions.

Future Work

While the proposed GA demonstrates promising results in generating feasible and lecturer-friendly course timetables, several avenues remain for future research and improvement. One potential direction is the integration of hybrid metaheuristics that combine GA with local search strategies such as Tabu Search or Simulated Annealing to enhance convergence speed and solution refinement. Such combinations may help escape local optima and produce more optimal schedules, especially in larger problems.

Another promising enhancement involves incorporating machine learning or reinforcement learning techniques to dynamically adjust penalty weights or predict lecturer preferences based on historical data. This adaptive approach could improve the algorithm's responsiveness to institutional scheduling patterns and policy or staff availability changes.

Additionally, future studies may involve feedback loops from faculty members or administrative staff to validate satisfaction levels more accurately and inform iterative model adjustments. Incorporating multi-objective optimization to balance different stakeholder priorities, such as room

utilization, lecturer satisfaction, and student timetable compactness, can also improve the model's real-world applicability.

Lastly, implementation in a live academic environment would offer valuable insights into deployment challenges, scalability, and user acceptance, forming a bridge between theoretical models and operational scheduling systems.

AUTHOR'S CONTRIBUTION

Khaeroni was the principal investigator, leading concept development, empirical data collection, and analysis. Additionally, Khaeroni was responsible for application design, manuscript drafting, and revisions to ensure the coherence and rigour of the study. Birru Muqdamien was critical in implementing the Genetic Algorithm using PHP, developing test units, and deploying the application to the server environment to ensure its functionality. Meanwhile, Ajeng Hesitiningtyas contributed to the composition of the introduction, literature review, and conclusion, focusing on model validation, methodological structuring, and the presentation of research findings. Furthermore, Ajeng Hesitiningtyas played a significant role in manuscript editing and refining the article to enhance its clarity, academic integrity, and overall feasibility.

COMPETING INTERESTS

Following the publication ethics of this journal, the authors—Khaeroni, Birru Muqdamien, and Ajeng Hesitiningtyas—hereby declare that this article is free from any conflict of interest (COI) or competing interests (CI) that could influence the integrity, objectivity, or validity of the research findings presented.

ACKNOWLEDGMENT

We want to express our deepest gratitude to the Research and Publication Center of the Research and Community Service Institute (LPPM) at UIN Sultan Maulana Hasanuddin Banten for providing research funding and support, enabling the successful completion of this study. Our sincere appreciation also goes to the Faculty of Education and Teacher Training at UIN Sultan Maulana Hasanuddin Banten for granting permission to conduct experiments and data collection, which contributed significantly to the development of the proposed model. Lastly, we thank the International Journal on Informatics for Development (IJID) for accepting and publishing this article.

REFERENCES

[1] K. Khairunnisa, "Penjadwalan Perkuliahan Otomatis," *FIBONACCI J. Pendidik. Mat. Mat.*, vol. 1, no. 1, pp. 1–14, 2015.

[2] E. Sugiarto, S. Winarno, and A. Fahmi, "Penjadwalan Perkuliahan Otomatis Berbasis Fuzzy Logic dan Genetic Algorithm pada Universitas Dian Nuswantoro," *Techno.COM J. Teknol. Inf.*, vol. 14, no. 4, pp. 315–328, 2015, doi: https://doi.org/10.33633/tc.v14i4.976.

[3] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A Honey-bee Mating Optimization Algorithm for Educational Timetabling Problems," *Eur. J. Oper. Res.*, vol. 216, no. 3, pp. 533–543, 2012, doi: https://doi.org/10.1016/j.ejor.2011.08.006.

[4] O. T. Arogundade, A. T. Akinwale, and O. M. Aweda, "A Genetic Algorithm Approach for a Real-World University Examination Timetabling Problem," *Int. J. Comput. Appl.*, vol. 12, no. 5, pp. 975 – 8887, Dec. 2010, doi: 10.5120/1678-2083.

[5] S. N. Jat and S. Yang, "A Memetic Algorithm for the University Course Timetabling Problem," in *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, 2008, vol. 1, pp. 427–433, doi: 10.1109/ICTAI.2008.126.

[6] S. A. Mirhassani, "A computational approach to enhancing course timetabling with integer programming," *Appl. Math. Comput.*, vol. 175, no. 1, pp. 814–822, 2006, doi: 10.1016/j.amc.2005.07.039.

[7] A. Darmawan and R. M. Hasibuan, "Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika dengan Mempertimbangkan Team-Teaching," in *Simposium Nasional RAPI XIII*, 2014, pp. 125–132.

[8] M. U. Devi, J. Jayapradha, B. Naharas, and S. Sharma, "Automated timetable generation for academic institutions," *AIP Conf. Proc.*, vol. 3075, no. 1, p. 20094, Jul. 2024, doi: 10.1063/5.0217280.

[9] S. Petrovic and E. Burke, "University Timetabling," in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, 1st ed., J. Y.-T. Leung, Ed. London: Chapman & Hall, 2004, pp. 45-1-45–19.

[10] T. Birbas, S. Daskalaki, and E. Housos, "School Timetabling for Quality Student and Teacher Schedules," *J. Sched.*, vol. 12, no. 2, p. 177, 2008, doi: 10.1007/s10951-008-0088-2.

[11] H. Kanoh and Y. Sakamoto, "Knowledge-based Genetic Algorithm for University Course Timetabling Problems," *KES J.*, vol. 12, pp. 283–294, Nov. 2008, doi: 10.3233/KES-2008-12403.

[12] H. Raoofpanah and V. Ghezavati, "Extended hybrid tabu search and simulated annealing algorithm for location-inventory model with multiple products, multiple distribution centers and multiple capacity levels," *Prod. Eng.*, vol. 13, no. 6, pp. 649–663, 2019, doi: 10.1007/s11740-019-00919-x.

[13] S. Bashath and A. R. Ismail, "Improved Particle Swarm Optimization by Fast Simulated annealing algorithm," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019, pp. 297–301, doi: 10.1109/ICAIIT.2019.8834515.

[14] A. O. Bajeh and K. O. Abolarinwa, "Optimization: A Comparative Study of Genetic and Tabu Search Algorithms," *Int. J. Comput. Appl.*, vol. 31, no. 5, pp. 975–8887, 2011.

[15] R. Gupta, P. Chadda, Y. K. Bhatt, A. Rawat, and G. Singh, "Genetic Algorithm-Based Timetable Creation System," in *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, 2024, vol. 1, pp. 1–6, doi: 10.1109/ICEECT61758.2024.10739290.

[16] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, 2005, doi: 10.1016/j.tcs.2005.05.020.

[17] H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, "A Review of Optimization Algorithms for University Timetable Scheduling," *Eng. Technol. Appl. Sci. Res.*, vol. 10, no. 6, pp. 6410–6417, 2020, doi: 10.48084/etasr.3832.

[18] A. I. Diveev and O. V. Bobr, "Variational Genetic Algorithm for NP-hard Scheduling Problem Solution," *Procedia Comput. Sci.*, vol. 103, no. October 2016, pp. 52–58, 2017, doi: 10.1016/j.procs.2017.01.010.

[19] E. A. Abdelhalim and G. A. El Khayat, "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA)," *Alexandria Eng. J.*, vol. 55, no. 2, pp. 1395–1409, 2016, doi: 10.1016/j.aej.2016.02.017.

[20] A. I. Diveev, O. V. Bobr, D. E. Kazaryan, and O. Hussein, "Some methods of solving the NP-difficult problem of optimal schedule for the university," *Procedia Comput. Sci.*, vol. 150, pp. 410–415, 2019, doi: 10.1016/j.procs.2019.02.071.

[21] F. Gerges, G. Zouein, and D. Azar, "Genetic algorithms with local optima handling to solve sudoku puzzles," in *ACM International Conference Proceeding Series*, 2018, pp. 19–22, doi: 10.1145/3194452.3194463.

[22] W. Stannat, "On the convergence of genetic algorithms - A variational approach," *Probab. Theory Relat. Fields*, vol. 129, no. 1, pp. 113–132, 2004, doi: 10.1007/s00440-003-0330-y.

[23] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimed. Tools Appl.*, vol. 80, pp. 8091–8126, 2021, doi: 10.1007/s11042-020-10139-6.

[24] A. Puspasari, K. Novianingsih, and F. Agustina, "Penyelesaian Masalah Penjadwalan Perkuliahan Menggunakan Algoritma Genetika (Studi Kasus Di Departemen Pendidikan Matematika Fpmipa Universitas Pendidikan Indonesia)," *J. EurekaMatika*, vol. 7, no. 1, pp. 80–92, 2019.

[25] L. Paranduk, A. Indriani, M. Hafid, and Suprianto, "Sistem Informasi Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika Berbasis Web," in *Seminar Nasional Aplikasi Teknologi Informasi (SNATi)*, 2018, pp. E46–E50.

[26] P. Khairunisak and Y. Hendriyani, "Aplikasi Penjadwalan Perkuliahan Menggunakan Algoritma Genetika (Studi Kasus : Jurusan Teknik Elektronika FT - UNP)," *Voteteknika J. Vocat. Tek. Elektron. dan Inform.*, vol. 9, no. 3, p. 24, 2021, doi: 10.24036/voteteknika.v9i3.112662.

[27] A. Qashlim and M. Assiddiq, "Penerapan Algoritma Genetika untuk Sistem Penjadwalan Kuliah," *J. Ilm. Ilmu Komput.*, vol. 2, no. 1, pp. 1–6, 2016.

[28] R. Hidayati, S. Bahri, and D. M. Midyanti, "Pengembangan Aplikasi Penjadwalan Matakuliah di Prodi Rekayasa Sistem Komputer Universitas Tanjungpura," *J. Tek. Inform. dan Sist. Inf.*, vol. 9, no. 3, pp. 2687–2698, 2022, doi: 10.35957/jatisi.v9i3.1499.

[29] D. W. Nugraha, A. Y. E. Dodu, and A. T. S. Saud, "Sistem Penjadwalan Perkuliahan Menggunakan Algoritma Genetika (Studi Kasus Pada Jurusan Teknologi Informasi Fakultas Teknik Universitas Tadulako)," *J. Ilm. Mat. dan Terap.*, vol. 14, no. 2, pp. 242–255, 2017, doi: 10.22487/2540766x.2017.v14.i2.9026.

[30] Y. Afandi and W. Setiyaningsih, "Sistem Pejadwalan Kuliah Meggunakan Metode Algoritma Genetika pada Program Magister Fakultas Ekonomi dan Bisnis," *RAINSTEK J. Terap. Sains Teknol.*, vol. 1, no. 1, pp. 40–47, 2019, doi: 10.21067/jtst.v1i1.3069.

[31] H. Ardiansyah and M. B. S. Junianto, "Penerapan Algoritma Genetika untuk Penjadwalan Mata Pelajaran," *J. Media Inform. Budidarma*, vol. 6, no. 1, p. 329, 2022, doi: 10.30865/mib.v6i1.3418.

[32] N. L. G. P. Suwirmayanti, I. M. Sudarsana, and S. Darmayasa, "Penerapan Algoritma Genetika Untuk Penjadwalan Mata Pelajaran," *J. Appl. Intell. Syst.*, vol. 1, no. 3, pp. 220–233, 2016.

[33] F. Mone and J. E. Simarmata, "Aplikasi Algoritma Genetika dalam Penjadwalan Mata Kuliah," *BAREKENG J. Ilmu Mat. dan Terap.*, vol. 15, no. 4, pp. 615–628, 2021, doi: 10.30598/barekengvol15iss4pp615-628.

[34] W. A. Puspaningrum, A. Djunaidy, and R. A. Vinarti, "Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika," *J. Tek. POMITS*, vol. 2, no. 1, pp. 153–160, 2013.

[35] D. C. Hajariwala, S. S. Patil, and S. M. Patil, "A Review of Metaheuristic Algorithms for Job Shop Scheduling," *Eng. Access*, vol. 11, no. 1, pp. 65–91, 2025, doi: 10.14456/mijet.2025.7.

[36] O. Maghiar *et al.*, "Behaviour Analysis of Trajectory and Population-Based Metaheuristics on Flexible Assembly Scheduling BT - Metaheuristics," in *Metaheuristics International Conference*, 2024, pp. 80–95, doi: 10.1007/978-3-031-62922-8_6.

[37] X. Weng, W. She, H. Fan, J. Zhang, and L. Yun, "Multi-depot vehicle routing problem with drones in emergency logistics," *Cluster Comput.*, vol. 28, no. 1, p. 64, 2024, doi: 10.1007/s10586-024-04809-5.

[38] L. Xiong and F. Yuan, "The Impact of Teacher Work Engagement on Student Engagement: Teaching Quality as a Mediator," *Soc. Behav. Personal. An Int. J.*, vol. 52, no. 9, pp. 1–8, 2024, doi: 10.2224/sbp.13541.

[39] A. Fikri Zaidan and L. Juariyah, "The Influence of Workloads on the Job Satisfaction of the Lecturers of State University of Malang Through Job Stress as Intervening Variable," in *International Conference on Islam, Economy, and Halal Industry*, 2020, vol. 2020, pp. 156–176, doi: 10.18502/kss.v4i9.7323.

[40] D. Lestari and Rahardianto, "Work Life Balance And Job Satisfaction Of Lecturer In Faculty Of Economics And Business Unjani," *Int. J. Sci. Technol. Manag.*, vol. 2, no. 5, pp. 1491–1504, 2021, doi: 10.46729/ijstm.v2i5.315.

[41] L. Kim, P. Pongsakornrungsilp, S. Pongsakornrungsilp, N. Horam, and V. Kumar, "Key Determinants of Job Satisfaction among University Lecturers," *Soc. Sci.*, vol. 12, no. 3, p. 153, 2023, doi: 10.3390/socsci12030153.

[42] E. Siahaan, P. Gultom, K. A. Fachrudin, and A. M. D. Sitohang, "Success Factors to Optimize the Job Satisfaction and Achieve Better Performance of Lecturers in Higher Education," *GATR Glob. J. Bus. Soc. Sci. Rev.*, vol. 10, no. 1, pp. 12–21, 2022, doi: 10.35609/gjbssr.2022.10.1(2).

[43] P. Sedgwick, "Non-parametric statistical tests for two independent groups: Numerical data," *BMJ*, vol. 348, no. February, pp. 9–11, 2014, doi: 10.1136/bmj.g2907.

[44] R. Chen, L. Su, and J. Guan, "AI-Powered Genetic Algorithm for Optimal Scheduling in Medical Research and Educational Management," *Comput. Aided. Des. Appl.*, vol. 21, no. S24, pp. 17–34, 2024, doi: 10.14733/cadaps.2024.S24.17-34.

[45] K. N. Subang, E. I. Balaba, and J. C. A. Jr, "Optimizing Course Scheduling with Genetic Algorithms : A Dynamic Approach," *SAR J.*, vol. 7, no. 4, pp. 296–302, 2024, doi: 10.18421/SAR74-02.

[46] K. L. Mills, J. J. Filliben, and A. L. Haines, "Determining relative importance and effective settings for genetic algorithm control parameters," *Evol. Comput.*, vol. 23, no. 2, pp. 309–342, 2015, doi: 10.1162/EVCO_a_00137.

[47] P. L. Iglesias, D. Mora, F. Javier Martinez, and V. S. Fuertes, "Study of sensitivity of the parameters of a genetic algorithm for design of water distribution networks," *J. Urban Environ. Eng.*, vol. 1, no. 2, pp. 61–69, 2007, doi: 10.4090/juee.2007.v1n2.061069.

[48] A. H. Khan and T. Imtiaz, "A Novel Genetic Algorithm Based Timetable Generator for Optimized University Timetable Solution," in *2024 International Conference on Engineering & Computing Technologies (ICECT)*, 2024, pp. 1–6, doi: 10.1109/ICECT61618.2024.10581296.

[49] M. A. Lopes, "A Metaheuristic Approach to Improving University Timetables Using Genetic Algorithms and Simulated Annealing," Universidade do Porto, 2024.

[50] S. Saraeian, "Sustainable Distributed Scheduling Problem," *Vietnam J. Comput. Sci.*, pp. 1–26, 2024, doi: 10.1142/S2196888824500222.

[51] C. C. Le, T. A. T. Nguyen, M. Q. Tran, and T. N. Phan, "Solving the Vietnamese EdTech Timetabling Problem with Optimized Multi-Objectives BT - Multi-disciplinary Trends in Artificial Intelligence," 2025, pp. 334–346.