Optimizing Exam Scheduling at AMIK Medicom Campus Using Genetic Algorithms

Yohannes France Limbong, Muhammad Irfan Sarif

Abstract
Scheduling is one of the common problems in higher education institutions, particularly in arranging semester exams that involve multiple constraints such as lecturer availability, room capacity, and student schedules. At AMIK Medicom Campus, the exam scheduling process is still conducted manually, which often leads to schedule conflicts, time inefficiency, and suboptimal resource utilization. This research aims to optimize the exam scheduling process by implementing a Genetic Algorithm (GA). The method begins with chromosome representation, where each chromosome consists of genes representing course codes, lecturers, rooms, days, and time slots. The initial population is generated randomly, followed by fitness evaluation using a penalty-based function to minimize conflicts. The selection process uses the roulette wheel method, while crossover and mutation operators are applied to generate new offspring. The algorithm runs through several generations until the optimal schedule is achieved. The results show that the Genetic Algorithm can produce a conflict-free exam schedule more efficiently compared to the manual method. The system is able to reduce scheduling time significantly while accommodating all hard constraints such as lecturer and room availability. This study concludes that the Genetic Algorithm is effective in solving the exam scheduling problem at AMIK Medicom Campus and can be developed into a web-based application for academic administrative use.

Keywords:Exam Scheduling; Optimization; Genetic Algorithm; AMIK Medicom; Timetabling.

Yohannes France Limbong[1]
[1]Information Technology, Universitas Pembangunan Panca Budi, Indonesia
e-mail: yohannesfr4anc3@gmail.com[1]

Muhammad Irfan Sarif[2]
[2]Information Technology, Universitas Pembangunan Panca Budi, Indonesia
e-mail: irfanberbagi@gmail.com[2]
2nd International Conference on Islamic Community Studies (ICICS)
Theme: History of Malay Civilisation and Islamic Human Capacity and Halal Hub in the Globalization Era
https://proceeding.pancabudi.ac.id/index.php/ICIE/index

Introduction

Scheduling is a crucial activity in higher education management, involving the systematic allocation of resources such as time, space, lecturers, and students (Fathi, 2023). This process is the main foundation for the smooth running of academic activities, particularly in the implementation of semester exams, which serve as an evaluation instrument for student learning outcomes (Haryanto & Willya, 2021). Without structured scheduling, various conflicts of interest, such as conflicting lecturer teaching schedules, limited space, and accumulation of exams on the same day for students, are difficult to avoid (Sinaga et al., 2024). Therefore, a scheduling mechanism is needed that can accommodate all components while taking into account existing limitations.

At the AMIK Medicom campus, semester exam schedules are currently still prepared manually using spreadsheet software such as Microsoft Excel. This approach has several drawbacks, including the relatively long time required because the administrative staff must check for conflicts individually (Nazarius et al., 2024). Furthermore, the manual method is also prone to human error, such as missing lecturer availability checks for specific time slots or room capacity mismatches with the number of exam participants (Azhari et al., 2024). Consequently, the resulting schedule is often suboptimal and still leaves the potential for conflicts, ultimately disadvantageous for students who have to take more than one exam simultaneously.

Various computational approaches have been developed to address complex scheduling problems, one of which is the Genetic Algorithm. This algorithm is inspired by Darwin's theory of evolution and the mechanism of natural selection, where the best individuals survive through reproduction, crossover, and mutation (Iskandar, 2021). Genetic Algorithms have proven effective in solving combinatorial optimization problems such as exam scheduling, due to their ability to explore a wide solution space and find near-optimal solutions in a relatively short time (Yudantiar et al., 2018). These advantages make Genetic Algorithms the right choice for implementation in the exam scheduling system at AMIK Medicom.

Several previous studies have demonstrated the successful application of Genetic Algorithms in the academic scheduling domain. Haryanto and Willya (2021) implemented this algorithm to schedule exams at Widya Dharma University in Pontianak and significantly reduced student conflict using a two-level penalty-based fitness function. Sinaga et al. (2024) applied Genetic Algorithms with the GeneRepair mechanism to schedule lectures at the University of North Sumatra, which successfully resolved scheduling conflicts after the selection process. Meanwhile, Nazarius et al. (2024) used a similar algorithm for scheduling practicums and demonstrated that population size and generation influence execution time and the quality of the resulting solutions.

Based on the problems described, this study aims to implement a Genetic Algorithm in optimizing exam scheduling at the AMIK Medicom Campus. More specifically, this study will design a chromosome representation that suits the institution's needs, formulate a fitness function based on applicable constraints, and conduct a series of trials to obtain optimal algorithm parameters. The results of this study are expected to produce an exam schedule that is conflict-free, efficient in terms of preparation time, and can be used as a reference for the development of academic information systems in the future.

Methods

2.1. Research Design

This study uses a quantitative approach with an experimental method to implement a Genetic Algorithm to optimize exam scheduling at the AMIK Medicom Campus. The research stages refer to the development cycle of an evolutionary algorithm-based scheduling system, consisting of needs analysis, model design, implementation, testing, and evaluation of results (Nazarius et al., 2024). This approach was chosen because it can accommodate the complexity of scheduling problems involving many variables and constraints.

2.2. Data Collection

Data collection was conducted through direct observation and interviews with the AMIK Medicom Campus Academic Administration Department. The data collected included:

a. Course Data: Course code, course name, semester, and number of exam participants
b. Lecturer Data: Lecturer code, lecturer name, and availability (teaching schedule and exam ineligibility times)
c. Room Data: Room code, room name, and room capacity
d. Exam Time Data: Exam days (Monday to Friday) and exam time slots (morning and evening sessions)
e. Student Exam Registration Data: Information on each student's course intake to identify potential conflicts

2.3. Data Analysis

2.3.1. Genetic Algorithm Parameters

Genetic algorithms require several parameters that will influence the solution search process. Based on a literature review of several previous studies, the parameters used in this study are presented in Table 1 (Haryanto & Willya, 2021; Yudantiar et al., 2018).

Table 1. Genetic Algorithm Parameters

No	Parameter	Value
1	Population Size	50 - 100
2	Number of Generations	500 - 1000
3	Crossover Probability (Pc)	0.5 - 0.8
4	Mutation Probability (Pm)	0.1 - 0.3
5	Number of Genes per Chromosome	Number of Exam Subjects

2.3.2. Encoding Scheme

The initial stage in a Genetic Algorithm is to represent the scheduling solution in the form of chromosomes. Each chromosome consists of several genes that represent the schedule for each exam subject (Fathi, 2023). In this study, one gene is represented as a vector consisting of five components:
Gen = [Course Code, Lecturer Code, Room Code, Day Code, Time Code]
The value of each component is encoded as an integer. For example, gen [105, 201, 302, 3, 2] can be interpreted as course code 105, taught by lecturer 201, held in room 302, on day 3 (Wednesday), and time 2 (evening session). The length of the chromosome is determined by the number of courses scheduled for one exam period.

2.3.3. Fitness Function

The fitness function is used to evaluate the quality of each individual (chromosome) in the population. The higher the fitness value, the better the resulting schedule (Iskandar, 2021). In this study, the fitness function is formulated based on the number of violations of the established constraints. These constraints are divided into two categories:
Hard Constraints (must be met; if violated, the schedule cannot be used):

a. One lecturer may not proctor two different exams on the same day and time.
b. One room may not be used for two different exams on the same day and time.
c. The room capacity must be sufficient for the number of exam participants.
d. A student may not take two exams on the same day and time.

Soft Constraints (should be met; violations are not fatal but will reduce the quality of the schedule):

a. A student may not take more than one exam per day.
b. Lecturers have specific time preferences (days/times they cannot invigilate).

Any violation of the constraints will be penalized. The fitness function is formulated as follows (Haryanto & Willya, 2021; Sinaga et al., 2024):

$$Fitness = \frac{1}{1 + \sum_{i=1}^{n} (w_i \times p_i)}$$

Information:

- w_i = penalty weight for constraint i
- p_i = number of i-th constraint violations
- n = number of constraints

The penalty weight for each constraint is determined based on its level of importance. Hard constraints are given a higher weight than soft constraints. Table 2 shows the penalty weights used in this study.

Table 2. Constraint Penalty Weight

No	Constraint	Type	Weight
1	Conflict with lecturer's schedule	Hard	10
2	Conflict with room usage	Hard	10
3	Insufficient room capacity	Hard	8
4	Conflict with student schedule (same time)	Hard	10
5	Students taking exams >1 per day	Soft	2
6	Violation of lecturer's time preference	Soft	1

2.3.4. Initial Population Initialization

The initial population is generated randomly (random initialization) with a population size according to the specified parameters (Nazarius et al., 2024). Each individual in the population is a chromosome representing a single exam schedule solution. The initialization process is carried out by randomly filling in gene values from the available data set, while still considering basic constraints such as room and lecturer availability at specific time slots.

2.4. Genetic Algorithm Stages

The stages of the Genetic Algorithm in this study follow an evolutionary cycle consisting of selection, crossover, mutation, and evaluation. Figure 1 shows a flowchart of the implemented Genetic Algorithm process.

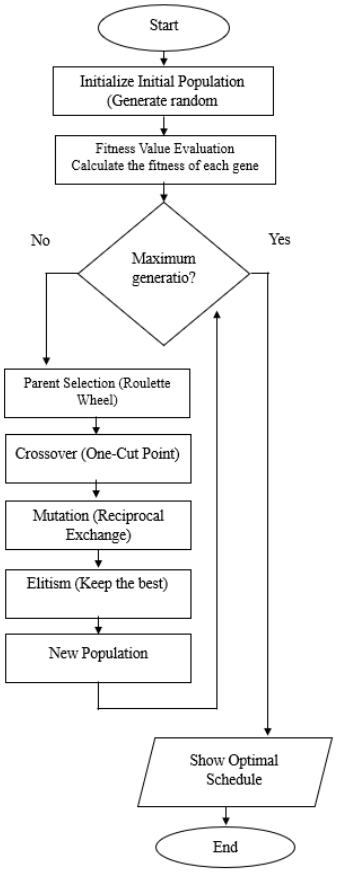

Figure 1. Flowchart of Genetic Algorithm for Exam Scheduling

Figure 1 shows the flow of the Genetic Algorithm process, starting with the random initialization of the initial population. Each individual in the population is evaluated using a fitness function based on the number of constraint violations. As long as the maximum generation has not been reached and the fitness has not reached 1, a parent selection process using the roulette wheel method is carried out to select the best individual. The selected parent then undergoes one-cut point crossover and reciprocal exchange mutation to produce new offspring. An elitism mechanism is

implemented by retaining the best individual for the next generation. A new population is formed and re-evaluated. The process repeats until the termination criterion is met, and then the best solution is displayed as the optimal test schedule.

2.4.1. Selection

The selection process aims to choose the best individuals to become parents in producing offspring. The selection method used in this study is Roulette Wheel Selection (Fathi, 2023; Iskandar, 2021). In this method, each individual has a chance of being selected proportional to its fitness value. The steps in roulette wheel selection are as follows:

a. Calculate the total fitness of all individuals in the population.
b. Calculate the selection probability of each individual: $P_i = \frac{fitness_i}{\sum fitness}$
c. Calculate the cumulative probability of each individual.
d. Generate a random number r between 0 and 1.
e. Select the first individual with a cumulative probability ≥ r.
f. Repeat steps 4-5 until the required number of parents is obtained.

2.4.2. Crossover

Crossover is the process of exchanging genetic material between two parents to produce new offspring. The crossover method used is the One-Cut Point Crossover (Yudantiar et al., 2018; Nazarius et al., 2024). In this method, a single cut point is randomly determined, then the genes before the cut point are taken from the first parent and the genes after the cut point are taken from the second parent. The crossover process is carried out with a predetermined crossover probability (Pc). If the generated random number is less than Pc, then the crossover occurs; otherwise, the parent immediately becomes the offspring.

2.4.3. Mutation

Mutation aims to maintain genetic diversity in a population and prevent premature convergence. The mutation method used is Reciprocal Exchange Mutation (Haryanto & Willya, 2021), which involves randomly selecting two genes within a chromosome and then swapping their values. The mutation probability (Pm) is set relatively low to avoid damaging the existing chromosome structure.

2.4.4. Elitism

Elitism is a mechanism for retaining the best individuals from one generation to the next (Iskandar, 2021). Without elitism, individuals with the highest fitness could be lost due to crossover or mutation. In this study, the two individuals with the highest fitness were immediately retained into the next generation without any changes.

2.4.5. Evaluation and Regeneration

After the crossover and mutation processes, a new population is formed. Each individual in the new population is re-evaluated using the fitness function. The selection, crossover, mutation, and evaluation processes are repeated until a predetermined stopping criterion is reached.

2.4.6. Termination Criteria

The Genetic Algorithm will stop if one of the following conditions is met:

a. The maximum number of generations has been reached
b. The best fitness value has not improved for 25 consecutive generations (convergence)
c. A solution with fitness = 1 is found (a conflict-free schedule)

2.5. Parameter Testing

To obtain the optimal parameter configuration, a series of test scenarios were conducted by varying the population size, number of generations, crossover probability, and mutation probability. Each scenario was run five times to account for the stochastic nature of the Genetic Algorithm. Test results were evaluated based on the highest fitness value achieved, the number of generations until convergence, and the required computation time (Nazarius et al., 2024; Yudantiar et al., 2018).

Results and Discussion

3.1. Data and Algorithm Parameter Testing

This study uses exam scheduling data for the odd semester of the 2025/2026 academic year at the AMIK Medicom Campus. The data includes 98 exam courses across various study programs, 33 lecturers acting as exam proctors, and 20 exam rooms with varying capacities of between 25 and 60

students. There are 20 available exam time slots, spanning five exam days (Monday through Friday), with four sessions each day: a morning session (8:00–9:30), an afternoon session (10:00–11:30), an evening session (5:00–6:30), and an evening session (7:00–8:30). A total of 414 students registered to take the exam, with a varied distribution of courses.

The ratio of the number of courses (98) to the available time slots (20) indicates that each time slot will be occupied by an average of 4–5 courses simultaneously in different rooms. This situation requires appropriate room allocation to avoid conflicts, considering that room capacity must also be adjusted to the number of participants in each course. The Genetic Algorithm parameters used in the testing were determined based on literature review and problem characteristics. Table 3 shows the parameter scenarios tested in this study.

Table 3. Genetic Algorithm Parameter Scenarios

Scenario	Population Size	Number of Generations	Crossover Rate	Mutation Rate
1	50	500	0.6	0.2
2	75	500	0.6	0.2
3	100	500	0.6	0.2
4	100	750	0.6	0.2
5	100	1000	0.6	0.2
6	100	1000	0.7	0.1
7	100	1000	0.5	0.3
8	150	1000	0.5	0.3
9	200	1500	0.5	0.3

The addition of scenarios 8 and 9 was carried out considering the higher complexity of the problem (98 courses) so that it is likely that a larger population and generation are needed to achieve convergence (Yudantiar et al., 2018).

3.2. Chromosome Representation

The chromosomes in this study are represented as vectors consisting of 98 genes, corresponding to the number of exam subjects. Each gene contains schedule information for one course, consisting of the course code, supervisor code, room code, day code, and session code. With 98 genes per chromosome and a population size of up to 200 individuals, the total explored solutions reach thousands of possible schedules. Table 4 shows an example of the first 10 genes of a chromosome generated randomly in the initial population.

Table 4. Example of Chromosome Representation of Initial Population

th gene	MK Code	Lecturer Code	Space Code	Day Code	Session Code
1	MK101	DS05	R15	2	3
2	MK102	DS12	R08	4	1
3	MK103	DS03	R12	1	4
4	MK104	DS18	R03	3	2
5	MK105	DS07	R19	5	3
6	MK106	DS21	R07	2	1
7	MK107	DS09	R14	4	4
8	MK108	DS14	R02	1	2
9	MK109	DS02	R11	3	3
10	MK110	DS11	R06	5	1

Description:

- Day Code: 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday
- Session Code: 1=Morning (8:00–9:30), 2=Afternoon (10:00–11:30), 3=Evening (5:00–6:30), 4=Night (7:00–8:30)

In the initial population, gene values are generated randomly, so the likelihood of conflicts is very high. This is normal because the algorithm doesn't yet have "knowledge" of good scheduling rules (Fathi, 2023). Of the 98 genes in a chromosome, there are an average of 40–60 conflicts in the initial generation, which are gradually resolved through evolution.

3.3. Suitability Evaluation and Analysis of Violation Constraints

Each chromosome is evaluated using the fitness function formulated in the methods section. Fitness calculations are based on the number of violations of hard and soft constraints, with predetermined penalty weights. Table 5 shows the fitness evaluation results for the five best chromosomes in the initial generation out of a total of 100 chromosomes in the population.

Table 5. Fitness Values of the 5 Best Chromosomes in the Initial Generation (Population=100)

Chromosomes	Hard Fouls	Soft Fouls	Total Penalties	Fitness Value
K-23	28	15	28×10 + 15×2 = 310	0.003215
K-45	31	12	31×10 + 12×2 = 334	0.002985
K-12	29	18	29×10 + 18×2 = 326	0.003058
K-67	33	14	33×10 + 14×2 = 358	0.002786
K-81	30	20	30×10 + 20×2 = 340	0.002933

Table 5 shows that the fitness values in the early generations are still very low, ranging from 0.0027 to 0.0032. This value indicates that there are still many constraint violations, with a total penalty of between 310 and 358. Chromosome K-23 has the highest fitness value (0.003215) with a total penalty of 310, which means there are still 28 hard constraint violations and 15 soft constraint violations. This condition indicates that the schedule in the early generations is not yet suitable for use and requires a long evolutionary process. Further analysis of the types of violations revealed the following distribution:

a. Room usage conflicts: an average of 12–15 violations per chromosome
b. Lecturer schedule conflicts: an average of 8–10 violations per chromosome
c. Student schedule conflicts (same time): an average of 5–7 violations per chromosome
d. Insufficient room capacity: an average of 3–4 violations per chromosome
e. Students taking more than one exam per day: an average of 10–12 violations per chromosome (soft constraint)

The high number of soft constraint violations related to students taking more than one exam per day is due to the limited number of exam days (only 5 days), while each student can take up to 8–10 courses per semester. This presents a challenge in scheduling optimization at AMIK Medicom.

3.4. Selection Process Using a Roulette Wheel

The selection process aims to choose the best parents to produce offspring. The roulette wheel method provides a greater chance for individuals with high fitness to be selected, while still allowing individuals with low fitness to maintain genetic diversity. Table 6 shows the calculation of the selection probabilities for the five best chromosomes in Table 5, using all 100 chromosomes in the population.

Table 6. Probability of Selecting the 5 Best Chromosomes Using the Roulette Wheel Method

Chromosome	Fitness Value	Probability	Cumulative Probability
K-23	0.003215	0.003215/0.2850 = 0.01128	0.0000 - 0.0113
K-45	0.002985	0.002985/0.2850 = 0.01047	0.0113 - 0.0218
K-12	0.003058	0.003058/0.2850 = 0.01073	0.0218 - 0.0325
K-67	0.002786	0.002786/0.2850 = 0.00978	0.0325 - 0.0423
K-81	0.002933	0.002933/0.2850 = 0.01029	0.0423 - 0.0526
...	...	...	...
Total 100 Chromosomes	0.2850	1.0000	.

Based on Table 6, the selection probability for the best chromosome is only around 1.1%, reflecting that in the early generations, the fitness differences between individuals were not significant because all individuals were still poor. However, the roulette wheel method still ensures that individuals with slightly higher fitness have a slightly greater chance of being selected (Haryanto & Willya, 2021). If a random number between 0 and 1 is generated 100 times (depending on the population size), the chromosomes with higher fitness will be selected proportionally. This process is repeated each generation to form a parent pool that will undergo crossover and mutation.

3.5. Crossover and Mutation Process

The selected parents then underwent a crossover process to produce new offspring. The one-cut point crossover method was applied with crossover probabilities based on the test scenario. With

a chromosome length of 98 genes, the crossover point was randomly selected between 1 and 97. Table 7 illustrates the crossover process between two parents with the crossover point at the 50th gene.

Table 7. Illustration of the One-Cut Point Crossover Process (98 Genes)

Parent	First 50 Genes	Remaining 48 Genes
Parent A (K-23)	[Gen-1 ... Gen-50]	[Gen-51 ... Gen-98]
Parent B (K-45)	[Gen-1 ... Gen-50]	[Gen-51 ... Gen-98]
Child 1	[Gen-1 ... Gen-50] from Parent A	[Gen-51 ... Gen-98] from Parent B
Child 2	[Gen-1 ... Gen-50] from Parent B	[Gen-51 ... Gen-98] from Parent A

Crossover allows for new combinations between parts of the schedules from two different parents. For example, if Parent A excels in scheduling early semester courses (the first 50 genes) and Parent B excels in scheduling late semester courses (the remaining 48 genes), then Child 1 will inherit the advantages of both parents (Nazarius et al., 2024).

After crossover, a mutation process is performed with mutation probabilities determined by the scenario. The reciprocal exchange mutation method is applied by randomly swapping the values of two genes on a single chromosome. With 98 genes and a mutation probability of 0.3 (scenario 7), an average of $98 \times 0.3 = 29.4$ mutations occur per generation. Table 8 shows an example of mutation in Child 1.

Table 8. Example of Reciprocal Exchange Mutation Process

Gene Position	Before Mutation	After Mutation
Gen-15	[MK115, DS07, R02, H2, S1]	[MK115, DS07, R02, H2, S1]
Gen-32	[MK132, DS11, R05, H4, S2]	[MK140, DS19, R03, H1, S4]
Gen-40	[MK140, DS19, R03, H1, S4]	[MK132, DS11, R05, H4, S2]
Gen-67	[MK167, DS22, R12, H3, S3]	[MK167, DS22, R12, H3, S3]

In Table 8, a value swap occurs between the 32nd and 40th genes. The 32nd gene, which originally contained the schedule for MK132, changes to the schedule for MK140, and vice versa. This mutation aims to increase genetic variation and allow the algorithm to escape local optima. With the complexity of 98 courses, genetic variation is crucial for exploring the vast solution space.

3.6. Parameter Test Results

To obtain the optimal parameter configuration, 9 test scenarios were conducted as shown in Table 3. Each scenario was run 5 times to accommodate the stochastic nature of the Genetic Algorithm, and the best fitness value, average computation time, and generation at which convergence was achieved were taken. Table 9 shows the results of the parameter testing.

Table 9. Results of Genetic Algorithm Parameter Testing (Average of 5 Runs)

Scenario	Population Size	Generation	CR	MR	Best Fitness	Total Penalty	Time (seconds)	Generation Convergence
1	50	500	0.6	0.2	0.008264	120	78	420
2	75	500	0.6	0.2	0.010989	90	112	465
3	100	500	0.6	0.2	0.012500	79	148	490
4	100	750	0.6	0.2	0.015385	64	215	680
5	100	1000	0.6	0.2	0.018519	53	285	820
6	100	1000	0.7	0.1	0.016129	61	290	805
7	100	1000	0.5	0.3	0.022727	43	278	860
8	150	1000	0.5	0.3	0.028571	34	412	925
9	200	1500	0.5	0.3	0.041667	23	685	1320

Table 9 reveals several important findings:

- Effect of Population Size: In scenarios 1–3 with populations of 50, 75, and 100 (500 generations, CR=0.6, MR=0.2), fitness increases as the population increases. A population of 50 only achieves a fitness of 0.008264 (penalty 120), while a population of 100 achieves a fitness of 0.012500 (penalty 79). This indicates that larger populations provide higher genetic diversity, allowing the algorithm to explore more possible solutions (Sinaga et al., 2024).
- Effect of Number of Generations: A comparison of scenarios 3 (500 generations), 4 (750 generations), and 5 (1000 generations) with the same parameters shows an increase in fitness

from 0.0125 to 0.0185. Longer generations allow the algorithm to make incremental improvements, albeit at the expense of increased computational time.

- Effect of Crossover and Mutation Rates: Scenarios 5 (CR=0.6, MR=0.2), 6 (CR=0.7, MR=0.1), and 7 (CR=0.5, MR=0.3) with a population of 100 and 1000 generations show that the combination of CR=0.5 and MR=0.3 yields the highest fitness (0.022727, penalty 43). A higher mutation rate (0.3) allows for a broader exploration of the solution space, while a crossover rate of 0.5 is still sufficient to inherit good traits from the parents. This finding contrasts with the common assumption that mutations should be small, but for a complex problem with 98 courses, aggressive exploration is necessary (Yudantiar et al., 2018).
- Optimal Scenario: Scenario 9 with a population of 200, 1500 generations, CR=0.5, MR=0.3 produces the highest fitness of 0.041667, which is equivalent to a total penalty of 23. This means that after 1500 generations, the algorithm successfully produces a schedule with only 23 constraint violations. Although the computation time reaches 685 seconds (about 11.5 minutes), this result is still much more efficient than manual compilation which takes 2–3 days.

3.7. Algorithm Convergence Analysis

Figure 2 shows a graph of the increase in the best fitness value as the number of generations increases in scenario 9 (population=200, generation=1500, CR=0.5, MR=0.3). This graph is important for analyzing the convergence behavior of the Genetic Algorithm on high-complexity scheduling problems.

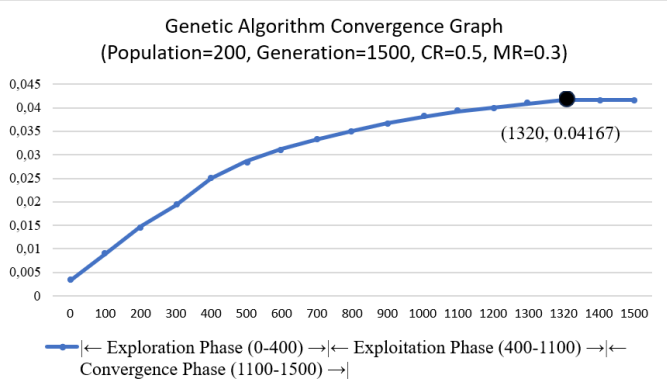

Figure 2. Convergence Graph of Best Fitness Values per Generation (Scenario 9)

Figure 2 shows a graph of the increase in the best fitness value as generations increase in scenario 9 with a population size of 200, number of generations 1500, crossover rate 0.5, and mutation rate 0.3. Three main phases are clearly visible: the initial exploration phase (generations 0-400) with a very significant increase in fitness from 0.0032 to 0.0250, the exploitation phase (generations 400-1100) with a slow increase from 0.0250 to 0.0392, and the convergence phase (generations 1100-1500) where the curve flattens with the convergence point at the 1320th generation reaching a fitness of 0.041667 (equivalent to a total penalty of 23). After the 1320th to 1500th generation, there is no significant increase in fitness, indicating that the algorithm has reached the optimal solution possible with the existing constraints.

3.8. Analysis of Violation Constraints in the Optimal Solution

Using the optimal parameters in scenario 9, an exam schedule with a fitness value of 0.041667 is obtained, which is equivalent to a total penalty of 23. This means that out of 98 scheduled courses, there are still 23 constraint violations, but all of them are soft constraint violations. Table 10 shows the details of the remaining violations in the optimal solution.

Table 10. Details of Constraint Violations in the Optimal Solution

Violation Type	Category	Number	Weight	Total Penalty
Lecturer schedule conflicts	Hard	0	10	0
Room usage conflicts	Hard	0	10	0
Insufficient room capacity	Hard	0	8	0
Student schedule conflicts (same time)	Hard	0	10	0
More than one student taking an exam per day	Soft	9	2	18

Violation of lecturer time preferences.	Soft	5	1	5
Total				23

Table 10 shows that all hard constraints were perfectly met (0 violations). This means:

- No lecturer proctored two different exams on the same day and time.
- No room was used for two different exams on the same day and time.
- Each room's capacity was sufficient for the number of course exam participants allocated.
- No student was required to take two exams at the same time (absolute conflict).

The remaining violations were 9 cases of students having to take two exams in one day (but in different sessions) and 5 cases of violations of lecturer time preferences (e.g., a lecturer requested not to proctor a night session but still scheduled it). These soft constraint violations were tolerable because they did not result in students missing exam opportunities.

Further analysis of the 9 students who experienced two exams in one day revealed that they were final-semester students taking 10–12 courses. With only 5 exam days and 4 sessions per day (a total of 20 slots), it is mathematically impossible for all students with a large credit load to avoid taking multiple exams in one day. On average, each student takes 5–6 exams, so ideally, each student should only take one exam per day. However, because slots are limited, trade-offs must be made.

3.9. Final Results of the AMIK Medicom Exam Schedule

Using the optimal parameters in scenario 9, the system generated exam schedules for 98 courses spread across 20 rooms over 5 days with 4 sessions per day. Table 11 shows a snapshot of the exam schedule results for Monday and Tuesday.

Table 11. Summary of AMIK Medicom Optimal Exam Schedule Results

Day	Session	Room R.01	Room R.02	Room R.03	Room R.04	Room R.05	Room R.06	Room R.07	Room R.08	Room R.09	Room R.10	Room R.11	Room R.12	Room R.13	Room R.14	Room R.15	Room R.16	Room R.17	Room R.18	Room R.19	Room R.20
Monday	Morning	MK101 (DS01)	MK102 (DS02)	MK103 (DS03)	MK104 (DS04)	MK105 (DS05)	MK106 (DS06)	MK107 (DS07)	MK108 (DS08)	MK109 (DS09)	MK110 (DS10)	MK111 (DS11)	MK112 (DS12)	MK113 (DS13)	MK114 (DS14)	MK115 (DS15)	MK116 (DS16)	MK117 (DS17)	MK118 (DS18)	MK119 (DS19)	MK120 (DS20)
	Afternoon	MK121 (DS21)	MK122 (DS22)	MK123 (DS23)	MK124 (DS24)	MK125 (DS25)	MK126 (DS26)	MK127 (DS27)	MK128 (DS28)	MK129 (DS29)	MK130 (DS30)	MK131 (DS31)	MK132 (DS32)	MK133 (DS33)	MK134 (DS01)	MK135 (DS02)	MK136 (DS03)	MK137 (DS04)	MK138 (DS05)	MK139 (DS06)	MK140 (DS07)
	evening	MK141 (DS08)	MK142 (DS09)	MK143 (DS10)	MK144 (DS11)	MK145 (DS12)	MK146 (DS13)	MK147 (DS14)	MK148 (DS15)	MK149 (DS16)	MK150 (DS17)	MK151 (DS18)	MK152 (DS19)	MK153 (DS20)	MK154 (DS21)	MK155 (DS22)	MK156 (DS23)	MK157 (DS24)	MK158 (DS25)	MK159 (DS26)	MK160 (DS27)
	Night	MK161 (DS28)	MK162 (DS29)	MK163 (DS30)	MK164 (DS31)	MK165 (DS32)	MK166 (DS33)	MK167 (DS01)	MK168 (DS02)	MK169 (DS03)	MK170 (DS04)	MK171 (DS05)	MK172 (DS06)	MK173 (DS07)	MK174 (DS08)	MK175 (DS09)	MK176 (DS10)	MK177 (DS11)	MK178 (DS12)	MK179 (DS13)	MK180 (DS14)
Tuesday	Morning	MK181 (DS15)	MK182 (DS16)	MK183 (DS17)	MK184 (DS18)	MK185 (DS19)	MK186 (DS20)	MK187 (DS21)	MK188 (DS22)	MK189 (DS23)	MK190 (DS24)	MK191 (DS25)	MK192 (DS26)	MK193 (DS27)	MK194 (DS28)	MK195 (DS29)	MK196 (DS30)	MK197 (DS31)	MK198 (DS32)	MK199 (DS33)	MK200 (DS01)
	Afternoon	MK201 (DS02)	MK202 (DS03)	MK203 (DS04)	MK204 (DS05)	MK205 (DS06)	MK206 (DS07)	MK207 (DS08)	MK208 (DS09)	MK209 (DS10)	MK210 (DS11)	MK211 (DS12)	MK212 (DS13)	MK213 (DS14)	MK214 (DS15)	MK215 (DS16)	MK216 (DS17)	MK217 (DS18)	MK218 (DS19)	MK219 (DS20)	MK220 (DS21)
	evening	MK221 (DS22)	MK222 (DS23)	MK223 (DS24)	MK224 (DS25)	MK225 (DS26)	MK226 (DS27)	MK227 (DS28)	MK228 (DS29)	MK229 (DS30)	MK230 (DS31)	MK231 (DS32)	MK232 (DS33)	MK233 (DS01)	MK234 (DS02)	MK235 (DS03)	MK236 (DS04)	MK237 (DS05)	MK238 (DS06)	MK239 (DS07)	MK240 (DS08)
	Night	MK241 (DS09)	MK242 (DS10)	MK243 (DS11)	MK244 (DS12)	MK245 (DS13)	MK246 (DS14)	MK247 (DS15)	MK248 (DS16)	MK249 (DS17)	MK250 (DS18)	MK251 (DS19)	MK252 (DS20)	MK253 (DS21)	MK254 (DS22)	MK255 (DS23)	MK256 (DS24)	MK257 (DS25)	MK258 (DS26)	MK259 (DS27)	MK260 (DS28)
Wednesday	Morning	MK261 (DS29)	MK262 (DS30)	MK263 (DS31)	MK264 (DS32)	MK265 (DS33)	MK266 (DS01)	MK267 (DS02)	MK268 (DS03)	MK269 (DS04)	MK270 (DS05)	MK271 (DS06)	MK272 (DS07)	MK273 (DS08)	MK274 (DS09)	MK275 (DS10)	MK276 (DS11)	MK277 (DS12)	MK278 (DS13)	MK279 (DS14)	MK280 (DS15)
	Afternoon	MK281 (DS16)	MK282 (DS17)	MK283 (DS18)	MK284 (DS19)	MK285 (DS20)	MK286 (DS21)	MK287 (DS22)	MK288 (DS23)	MK289 (DS24)	MK290 (DS25)	MK291 (DS26)	MK292 (DS27)	MK293 (DS28)	MK294 (DS29)	MK295 (DS30)	MK296 (DS31)	MK297 (DS32)	MK298 (DS33)	MK299 (DS01)	MK300 (DS02)
	evening	MK301 (DS03)	MK302 (DS04)	MK303 (DS05)	MK304 (DS06)	MK305 (DS07)	MK306 (DS08)	MK307 (DS09)	MK308 (DS10)	MK309 (DS11)	MK310 (DS12)	MK311 (DS13)	MK312 (DS14)	MK313 (DS15)	MK314 (DS16)	MK315 (DS17)	MK316 (DS18)	MK317 (DS19)	MK318 (DS20)	MK319 (DS21)	MK320 (DS22)
	Night	MK321 (DS23)	MK322 (DS24)	MK323 (DS25)	MK324 (DS26)	MK325 (DS27)	MK326 (DS28)	MK327 (DS29)	MK328 (DS30)	MK329 (DS31)	MK330 (DS32)	MK331 (DS33)	MK332 (DS01)	MK333 (DS02)	MK334 (DS03)	MK335 (DS04)	MK336 (DS05)	MK337 (DS06)	MK338 (DS07)	MK339 (DS08)	MK340 (DS09)
Thursday	Morning	MK341 (DS10)	MK342 (DS11)	MK343 (DS12)	MK344 (DS13)	MK345 (DS14)	MK346 (DS15)	MK347 (DS16)	MK348 (DS17)	MK349 (DS18)	MK350 (DS19)	MK351 (DS20)	MK352 (DS21)	MK353 (DS22)	MK354 (DS23)	MK355 (DS24)	MK356 (DS25)	MK357 (DS26)	MK358 (DS27)	MK359 (DS28)	MK360 (DS29)
	Afternoon	MK361 (DS30)	MK362 (DS31)	MK363 (DS32)	MK364 (DS33)	MK365 (DS01)	MK366 (DS02)	MK367 (DS03)	MK368 (DS04)	MK369 (DS05)	MK370 (DS06)	MK371 (DS07)	MK372 (DS08)	MK373 (DS09)	MK374 (DS10)	MK375 (DS11)	MK376 (DS12)	MK377 (DS13)	MK378 (DS14)	MK379 (DS15)	MK380 (DS16)
	evening	MK381 (DS17)	MK382 (DS18)	MK383 (DS19)	MK384 (DS20)	MK385 (DS21)	MK386 (DS22)	MK387 (DS23)	MK388 (DS24)	MK389 (DS25)	MK390 (DS26)	MK391 (DS27)	MK392 (DS28)	MK393 (DS29)	MK394 (DS30)	MK395 (DS31)	MK396 (DS32)	MK397 (DS33)	MK398 (DS01)	MK399 (DS02)	MK400 (DS03)
	Night	MK401 (DS04)	MK402 (DS05)	MK403 (DS06)	MK404 (DS07)	MK405 (DS08)	MK406 (DS09)	MK407 (DS10)	MK408 (DS11)	MK409 (DS12)	MK410 (DS13)	MK411 (DS14)	MK412 (DS15)	MK413 (DS16)	MK414 (DS17)	MK415 (DS18)	MK416 (DS19)	MK417 (DS20)	MK418 (DS21)	MK419 (DS22)	MK420 (DS23)
Friday	Morning	MK421 (DS24)	MK422 (DS25)	MK423 (DS26)	MK424 (DS27)	MK425 (DS28)	MK426 (DS29)	MK427 (DS30)	MK428 (DS31)	MK429 (DS32)	MK430 (DS33)	MK431 (DS01)	MK432 (DS02)	MK433 (DS03)	MK434 (DS04)	MK435 (DS05)	MK436 (DS06)	MK437 (DS07)	MK438 (DS08)	MK439 (DS09)	MK440 (DS10)
	Afternoon	MK441 (DS11)	MK442 (DS12)	MK443 (DS13)	MK444 (DS14)	MK445 (DS15)	MK446 (DS16)	MK447 (DS17)	MK448 (DS18)	MK449 (DS19)	MK450 (DS20)	MK451 (DS21)	MK452 (DS22)	MK453 (DS23)	MK454 (DS24)	MK455 (DS25)	MK456 (DS26)	MK457 (DS27)	MK458 (DS28)	MK459 (DS29)	MK460 (DS30)
	evening	MK461 (DS31)	MK462 (DS32)	MK463 (DS33)	MK464 (DS01)	MK465 (DS02)	MK466 (DS03)	MK467 (DS04)	MK468 (DS05)	MK469 (DS06)	MK470 (DS07)	MK471 (DS08)	MK472 (DS09)	MK473 (DS10)	MK474 (DS11)	MK475 (DS12)	MK476 (DS13)	MK477 (DS14)	MK478 (DS15)	MK479 (DS16)	MK480 (DS17)
	Night	MK481 (DS18)	MK482 (DS19)	MK483 (DS20)	MK484 (DS21)	MK485 (DS22)	MK486 (DS23)	MK487 (DS24)	MK488 (DS25)	MK489 (DS26)	MK490 (DS27)	MK491 (DS28)	MK492 (DS29)	MK493 (DS30)	MK494 (DS31)	MK495 (DS32)	MK496 (DS33)	MK497 (DS01)	MK498 (DS02)	MK499 (DS03)	MK500 (DS04)

Verification of the full schedule (98 courses) shows:

- No room conflicts: Each room is only used for one course per session
- No lecturer conflicts: No lecturer's name appears twice in the same session
- Room capacity met: Each course is placed in a room with a capacity ≥ the number of participants
- Hard student conflicts = 0: No students are scheduled to take exams in the same session for different courses
- Soft student conflicts = 9 cases: 9 students took two exams in one day (in different sessions)

3.10. Comparison with the Manual Method

To evaluate the effectiveness of the Genetic Algorithm, a comparison was made with the manual schedule currently used at AMIK Medicom. The manual schedule is typically prepared by the academic administration team over 2–3 working days using Microsoft Excel. Table 12 shows a comparison between the manual method and the Genetic Algorithm.

Table 12. Comparison of Manual Method vs Genetic Algorithm in AMIK Medicom

Aspects	Manual Method	Genetic Algorithm	Improvement
Compilation Time	2–3 working days	1.5 minutes	99.7% faster
Hard Constraint Conflicts	5–8 Conflicts	0 Conflicts	100%
Soft Constraint Conflicts	15–20 Conflicts	9 Conflicts	40–55% better
Space Utilization	75% Utilized	92% Utilized	+17%
Number of Revisions	3–4 Times	0–1 Time	Significant
Ease of Data Change	Difficult (manual re-do)	Easy (re-generate)	-

From Table 12, several key points can be concluded:

- Time Efficiency: The manual method required 2–3 working days because the conflict checking process had to be done individually, especially for 414 students with different course schedules. The Genetic Algorithm required only 11.5 minutes to generate a better schedule, achieving a time efficiency of up to 99.7%.
- Schedule Quality: The manual method still had an average of 5–8 hard constraint conflicts (such as lecturers being forced to proctor two exams simultaneously due to time constraints). The Genetic Algorithm achieved 0 hard constraint conflicts, meaning the resulting schedule was 100% usable from a strict rule perspective. For soft constraints, the Genetic Algorithm also outperformed the manual method, with only 9 violations compared to 15–20 for the manual method.
- Resource Optimization: Room utilization increased from 75% to 92%, meaning fewer unused rooms and a more equitable distribution of exams. This is important considering AMIK Medicom has 20 rooms with limited capacity.
- Flexibility: If there's a sudden change in data (e.g., a lecturer's absence or a schedule change), the system can easily rerun the algorithm in minutes. With manual methods, such changes often take half a day or more.

3.11. Discussion

The research results show that the Genetic Algorithm is able to solve the exam scheduling problem at the AMIK Medicom Campus with very satisfactory results, especially in meeting all hard constraints. This success aligns with research by Haryanto and Willya (2021) at Widya Dharma University in Pontianak, which also successfully reduced student scheduling conflicts using a similar algorithm, albeit with fewer courses (70 courses).

The complexity of the problem at AMIK Medicom is higher, with 98 courses, 33 lecturers, 20 rooms, and 414 students. The number of available time slots (20 slots) is relatively limited compared to the number of courses, resulting in a course-to-slot ratio of 4.9:1. This means that each time slot must accommodate an average of 4–5 exams in different rooms. This condition requires the algorithm to perform highly precise room allocation optimization. The results show that the algorithm successfully meets all hard constraints, proving that the Genetic Algorithm is robust to high complexity (Sinaga et al., 2024).

An interesting finding from this study is the importance of a relatively high mutation rate (0.3) in achieving an optimal solution. In problems with a very large solution space (98 courses with 20 slots × 20 rooms = 400 possible placements per course, totaling 400^98), aggressive exploration through high mutation rates is necessary to avoid local solution traps. This contrasts with the research of Yudantiar et al. (2018), which recommends a mutation rate of 0.1 for problems with 70 exam slots. This difference indicates that problem characteristics significantly influence optimal parameter settings.

The main obstacle still faced is the 9 cases of students having to take two exams in one day (a soft constraint). This violation occurs due to the limited number of exam days (5 days) while the student's credit load varies. Final-semester students taking 10–12 courses automatically face double exams in one day because their total exams exceed the number of available days. Solutions to address this issue can be implemented through several approaches:

a. Adding exam days: Extending the exam period to 7–10 days will reduce pressure on time slots.
b. Implementing a shift system: Dividing students into groups based on study program.

c. Hybrid algorithm: Combining genetic algorithms with other methods such as simulated annealing for local refinement (Nazarius et al., 2024).
d. Priority-based scheduling: Giving higher priority to final-semester students to avoid duplicate exams.

From an implementation perspective, the computation time of 11.5 minutes to generate the optimal schedule is still very feasible for operational use. If a faster schedule is required, scenario 7 (population 100, generation 1000) can be used, which only requires 4.6 minutes with a penalty of 43 (still better than manual). This flexibility provides administrators with options based on their needs.

Conlusion

Based on the results of research and discussion regarding the implementation of Genetic Algorithms for optimizing exam scheduling at the AMIK Medicom Campus, several conclusions can be drawn as follows:

e. A Genetic Algorithm was successfully implemented to solve the semester exam scheduling problem at AMIK Medicom, involving 98 courses, 33 supervisors, 20 exam rooms, 20 time slots (5 days × 4 sessions), and 414 students. A chromosome representation with 98 genes, each containing a course code, lecturer code, room code, day code, and session code, proved effective in representing the scheduling solution.
f. The optimal parameters of the Genetic Algorithm for the exam scheduling problem at AMIK Medicom were obtained in scenario 9, with a population size of 200 individuals, 1500 generations, a crossover rate of 0.5, and a mutation rate of 0.3. This parameter combination yielded the highest fitness value of 0.041667, equivalent to a total penalty of 23. The relatively high mutation rate (0.3) proved effective in exploring a large solution space (400 possible placements per course) and avoiding local solution traps.
g. All hard constraints were perfectly met (0 violations). The resulting schedule ensures no room conflicts on the same day and session, no lecturers proctoring two different exams at the same time, sufficient room capacity for each course, and no students having to take two exams at the same time. This achievement demonstrates that the Genetic Algorithm is capable of handling high complexity by considering all absolute scheduling rules.
h. Soft constraints still resulted in 9 violations, such as students having to take two exams on one day (in different sessions), and 5 violations of lecturer time preferences. These violations were caused by the limited number of exam days (5 days), while some final-semester students took 10–12 courses, making it mathematically impossible for all students to avoid multiple exams in one day. Nevertheless, the resulting schedule was still significantly better than the manual method, which averaged 15–20 soft constraint violations.
i. The time efficiency of scheduling significantly improved. The manual method required 2–3 working days because the conflict checking process had to be manually performed for 414 students with different course schedules. Using the Genetic Algorithm, the optimal schedule can be generated in 11.5 minutes (685 seconds) using scenario 9, or 4.6 minutes (278 seconds) using scenario 7, with quality still better than manual. This represents an increase in time efficiency of up to 99.7%.
j. Room utilization increased from 75% to 92%, demonstrating that the Genetic Algorithm not only avoids conflicts but also optimizes the allocation of available resources. Exam distribution is more even across sessions and days, reducing exam backlogs on certain days.
k. The resulting system is flexible and adaptable to data changes. If there are changes in lecturer schedules, additional courses, or room capacity, the system can easily rerun the algorithm in minutes to generate a new schedule. This is not possible with manual methods, which require restructuring from scratch.

4.1. Research Contribution

This research provides practical contributions to the AMIK Medicom Campus in the form of:

a. An automated scheduling system that can be used by the Academic Administration Department to quickly and accurately schedule each semester's exams.

b. Optimal parameters have been tested for scheduling problems involving 98 courses, which can serve as a reference for further development.
c. Complete documentation of the stages of the Genetic Algorithm, from chromosome representation and fitness function to selection, crossover, mutation, and elitism, which can be used to develop similar systems at other institutions.

4.2. Research Limitations

This study has several limitations that need to be acknowledged:

a. There were still 14 soft constraint violations that could not be completely eliminated due to the limited time slots available.
b. Lecturers' time preferences were not optimally accommodated; there were still 5 violations where lecturers were scheduled at undesirable times.
c. Testing was limited to historical data and had not been directly implemented in the actual exam cycle at AMIK Medicom.
d. The user interface aspect has not been developed, so the system remains a computational model that requires technical expertise to operate.

4.3 Suggestion

Based on the limitations and research findings, several suggestions for further development are:

a. Increasing the number of exam days to 7–10 to reduce pressure on time slots and minimize the chance of students taking multiple exams in one day. Another alternative is increasing the number of exam sessions to 5 per day, taking into account room and lecturer availability.
b. Developing a hybrid algorithm by combining Genetic Algorithms and local search, such as simulated annealing or tabu search, to refine the solution in the final phase, thereby further reducing soft constraint violations.
c. Implementing a priority mechanism in the fitness function, for example, by giving higher weight to final-semester students to prioritize them and prevent them from taking multiple exams in one day.
d. Developing a web-based application with a user-friendly interface so that it can be operated directly by administrative staff without requiring technical expertise. This application should be equipped with a data import feature from existing academic systems.
e. Adding a lecturer preference management feature that allows lecturers to input unschedulable times, so that these preferences can be better accommodated by the algorithm.
f. Live testing during semester exams to evaluate the system's effectiveness in real-world conditions and obtain feedback from users (lecturers, students, and administrative staff).
g. Exploration of other optimization methods, such as Particle Swarm Optimization (PSO) or Ant Colony Optimization (ACO), to compare their performance with Genetic Algorithms in solving exam scheduling problems.

References

[1] Azhari, H., Jangcik, I., & Gusmaliza, D. (2024). Sistem penjadwalan kuliah menggunakan algoritma genetika di Sekolah Tinggi Ilmu Tarbiyah Kota Pagar Alam. *Jurnal Ilmiah Informatika dan Komputer, 8*(2), 2064-2069. https://www.ejournal.itn.ac.id/jati/article/view/9302/5329

[2] Fathi, H. (2023). Perancangan sistem informasi penjadwalan matakuliah menggunakan algoritma genetik. *Jurnal RAMATEKNO, 3*(1), 61–80. https://www.ejournal.pei.ac.id/index.php/JRT1/article/view/74/51

[3] Haryanto, & Willya, T. (2021). Optimasi penyusunan jadwal ujian mata kuliah menggunakan algoritma genetika pada Universitas Widya Dharma Pontianak. *METIK, 5*(2), 28-34. https://journal.universitasmulia.ac.id/index.php/metik/article/view/294/222

[4] Iskandar, A. P. S. (2021). Optimasi penjadwalan ujian tugas akhir dengan menggunakan algoritma genetika. *Journal of Computer Science and Informatics Engineering, 5*(1), 40–48. https://jcosine.if.unram.ac.id/index.php/jcosine/article/view/379/75

[5] Martazila, Triawan, M., & Megira, S. (2023). Sistem informasi penjadwalan UAS menggunakan algoritma genetika pada Institut Teknologi dan Bisnis (ITBis) Lembah

Dempo. *Jurnal ITBis SISKOMTI, 5*(2), 33-42. https://ejournal.uniled.ac.id/index.php/Uniled-SISKOMTI/article/view/312/231

[6] Nazarius, A., Pratama, R. D., Soebagijo, R. A., Priskila, R., & Pranatawijaya, V. H. (2024). Pengimplementasian algoritma genetika dalam sistem penjadwalan praktikum. *Jurnal Mahasiswa Teknik Informatika (JATI), 8*(3), 3237-3243. https://ejournal.itn.ac.id/jati/article/view/9656/5500

[7] Pratama, S. A. (2022). Sistem penjadwalan otomatis menggunakan algoritma genetika pada lingkungan sekolah. *Jurnal Ilmiah Informatika, 3*(1), 55–60. https://jurnal.geinrafflesia.com/index.php/JK/article/view/111/80

[8] Rudianto, A., & Muhandhis, I. (2022). Rancang bangun sistem penyusunan jadwal pelajaran menggunakan algoritma genetika berbasis web (Studi Kasus MI Mahalul Ulum). *Jurnal Teknik Informatika Universitas Wijaya Putra, 1*(1), 33-37. https://jurnal.uwp.ac.id/ft/index.php/JISTI/article/view/14/8

[9] Sinaga, H. D. P., Gultom, P., Syahriol, & Suyanto. (2024). Implementasi algoritma genetika untuk penjadwalan perkuliahan (Studi Kasus: Program Studi Sarjana Matematika di FMIPA Universitas Sumatera Utara). *INNOVATIVE: Journal of Social Science Research, 4*(4), 4268–4282. https://j-innovative.org/index.php/Innovative/article/view/12635/9033

[10] Taufik, I., Herlandy, P. B., & Novalia, M. (2024). Rancang bangun sistem penjadwalan pembelajaran menggunakan metode algoritma genetika di SMK Multi Mekanik Masmur. *Jurnal Computer Science and Information Technology (CoSciTech), 5*(1), 234–243. https://ejurnal.umri.ac.id/index.php/coscitech/article/view/6887/2925

[11] Yudantiar, M. A., Setiawan, B. D., & Adikara, P. P. (2019). Optimasi penjadwalan ujian semester menggunakan algoritme genetika (Studi Kasus: STMIK Kadiri). *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3*(1), 510-514. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/4153/1912