Vol. 7, No. 1, 2026, pp. 2197-2210
DOI: https://doi.org/10.29210/994320

Contents lists available at Elora Center

Lentera Negeri

Journal homepage: https://lentera.eloracenter.org/lentera



Hybrid genetic algorithm with dsatur seeding for automated course timetabling: a three-stage comparative study

Ary Ramadhan[*)1], Rifqi Aulia Zaim[1], Aditya Hanum Widarsa[1], Maltha Kharisma[2], Arif Budiman[1], Vivi Indriyani[3], Alrizka Hairi Dilfa[4]

[1] Desain Komunikasi Visual, Fakultas Bahasa dan Seni, Universitas Negeri Padang, Indonesia
[2] Pendidikan Seni Rupa, Fakultas Bahasa dan Seni, Universitas Negeri Padang, Indonesia
[3] Pendidikan Bahasa Indonesia, Fakultas Bahasa dan Seni, Universitas Negeri Padang, Indonesia
[4] Pendidikan Musik, Fakultas Bahasa dan Seni, Universitas Negeri Padang, Indonesia

Article Info

Article history:
Received Sep 01th, 2026
Revised Oct 15th, 2026
Accepted Oct 29th, 2026

Keyword:
Genetic algorithm,
DSATUR,
University course timetabling,
Room-type differentiation,
Hybrid algorithm

ABSTRACT

This study aims to develop and evaluate a web-based university course timetabling system that hybridizes a Genetic Algorithm (GA) with DSATUR graph-coloring seeding, and to quantify the separate effects of room-type constraints and DSATUR seeding on hard-constraint satisfaction and convergence. The system was developed using the 4D model (Define, Design, Develop, Disseminate) and implemented in the CodeIgniter framework. Real data from the odd semester of the 2025/2026 academic year of the Visual Communication Design Study Program, Universitas Negeri Padang, were used, comprising 87 class sections, 26 courses, 16 lecturers, 12 theory rooms, and 6 practicum studios. A conflict was operationally defined as a lecturer or a room assigned to more than one class section in the same time slot, and a run was successful when it reached the maximum fitness of 87 with zero conflicts. Three stages were compared over ten independent runs each: a basic GA (Stage 1), a GA with room-type separation (Stage 2), and a GA with room-type separation and a 10% DSATUR-seeded initial population (Stage 3). Differences were tested using Fisher's exact test and Welch's t-test, with Cohen's d as the effect size. Room-type separation increased the success rate from 10% to 100% ($p < 0.001$) and reduced the mean convergence generation from 405.9 to 174.4 ($p < 0.001$). DSATUR seeding further reduced the mean convergence generation by 15.5% (from 174.4 to 147.3) and the mean computation time by 9.8% (from 71.16 s to 64.22 s); however, these reductions were not statistically significant with ten runs ($p = 0.372$ and $p = 0.132$). The findings show that domain-specific constraint modeling is the decisive factor for feasibility, whereas heuristic seeding provides a moderate efficiency gain. The final system produced conflict-free timetables in every run and reduced a manual process of several days to about one minute of computation.

© 2026 The Authors.
This is an open access article under the CC BY-NC-SA license (https://creativecommons.org/licenses/by-nc-sa/4.0)

Corresponding Author:
Ary Ramadhan,
aryramadhan@fbs.unp.ac.id

Introduction

University course timetabling problem (UCTP) is a combinatorial optimization issue, categorized as one of NP-hard problems [1][2][3][4][5]. The complexity increases rapidly when the number of courses, lecturers, rooms, and time segments available for assignment grows. Constraints in timetabling are divided generally

into hard constraints (i.e., those that must be met, like the one that a lecturer cannot be teaching two classes simultaneously) and soft constraints (which, when feasible, should be met such as lecturers' preferred times) [6]. Art and design education at the higher level brings even more complexity because the hands-on practice course requires dedicated studios or labs that regular lecture rooms cannot replace [7][8].

Until recently, the Study Program of Visual Communication Design (VCD) at Universitas Negeri Padang prepared its timetables manually using spreadsheet software, a practice also reported in other institutional case studies [9]. For one semester, 87 class sections must be assigned to 18 rooms (12 theory rooms and 6 practicum studios) and 50 weekly time slots, so each section has up to $18 \times 50 = 900$ candidate room–time combinations and the search space grows to the order of 90087 possible timetables. Under these conditions, manual preparation takes several working days, and every local adjustment can create new conflicts, such as overlapping lecturer schedules, the same room being used by two classes at the same time, or theory rooms being used for practicum teaching. An automated solution is therefore needed that also respects the specific facility constraints of a studio-based program.

The Genetic Algorithm (GA), a type of metaheuristic technique, was inspired by the process of natural evolution to the point where it has become an effective tool in solving a great variety of scheduling problems [10][11]. The GA encodes possible problem solutions as strings of genes (chromosomes), where each gene represents an element of a solution. The GA works through a combination of selection, crossover, and mutation processes where the fittest solutions (chromosomes) are more likely to survive and produce offspring [12][13][14][15]. The GA can be used for searching an optimal solution in a very large search space; however, its performance is primarily affected by the quality of the initial population [16]. Random initial population generation usually results in a very long convergence period and very inconsistent quality of the final solution [17]. The GA was selected over single-solution metaheuristics such as simulated annealing and tabu search [31][33] for two reasons: its population-based search can directly accept several heuristic seed solutions, which is the mechanism investigated in this study, and its chromosome encoding maps naturally onto section-level assignments of lecturers, rooms, and time slots [14][15].

Graph coloring gives a different way to look at timetabling [18] [19]. By using a graph, you can represent a timetabling problem, where the vertices are the class sections, the edges are the conflicts and the colors are the time slots. The DSATUR (saturation degree) algorithm is a constructive heuristic whereby the algorithm goes through coloring the nodes according to their degree of saturation which means the number of colors that the nearby (neighboring) nodes have already been colored with. DSATUR is recognized to produce colorings that are close to the optimal ones and it has negligible computational overhead [20][21][22].

Several studies have combined the GA with graph coloring or other constructive heuristics, for example GA analysis with graph-coloring operators [29], Welsh–Powell coloring for course scheduling [21], GA with local search [17], and hyper-heuristic or memetic frameworks [23][27]; web-based GA timetabling systems have also been reported [24][26]. However, these studies generally treat all rooms as interchangeable and evaluate the hybrid only as a whole, so the separate contribution of domain constraints and heuristic seeding remains unknown. Moreover, studio- and laboratory-based programs in art and design education, in which practicum courses must be held in dedicated studios, have rarely been examined [7][8]. This study addresses this gap by combining domain-based room-type partitioning with DSATUR seeding of the GA population in a stepwise (ablation) design.

This paper introduces a GA-DSATUR hybrid algorithm that combines the best features of both. DSATUR generates well-built correct initial solutions that are, in turn, plugged into the initial GA population to get a faster convergence and higher consistency results. Two essential constraints that the authors found are usually neglected in research are implemented in the designed system: 1) a contiguity constraint that makes sure a course's time slot schedule is continuous the same day depending on its credit load; and 2) room-type differentiation, i.e., theory rooms vs. practicum rooms, which makes the distribution strictly separate.

The major contribution of our work is a comprehensive study of how algorithmic improvements translate into results by conducting three different experiments. In the first stage, the performance of a simple GA without the room type separation feature is measured as a baseline. In the second stage, we check the performance of the system with the room type constraint only. Then, in the third experiment, we introduce the concept of DSATUR seeding into the system which has been already equipped with room type separation. These three stages enable us to precisely evaluate the impact of each change, an ablation design that is rarely applied in the course timetabling literature.

The objectives of this study are threefold: (1) to develop and implement a web-based timetabling system based on a hybrid GA–DSATUR algorithm that enforces credit-contiguity and room-type constraints; (2) to measure, through a three-stage ablation experiment, the effect of room-type separation on the success rate and the number of hard-constraint conflicts; and (3) to test, using inferential statistics, whether DSATUR seeding reduces the convergence generation and computation time compared with a randomly initialized GA. The novelty of this work lies in quantifying separately the contributions of domain constraint modeling and heuristic seeding in a studio-based art and design program.

Method

System Development Model

This study applied the Research and Development (R&D) method using the 4D development model, which comprises four stages [25]. In the Define stage, requirements were gathered through semi-structured interviews with the Head of the Study Program and the administrative staff responsible for preparing the timetable. These informants were selected purposively because they are the only parties directly involved in producing and approving the semester timetable, and their statements were triangulated with the previous semesters' spreadsheet timetables and the official course–lecturer assignment documents. The main requirements identified were: (a) the system must produce conflict-free timetables; (b) teaching time slots must be consecutive within the same day according to the course credit load; and (c) practicum courses must be limited to studios or laboratories. The Design stage covered the system architecture, the relational database schema, the user interface, and the structure and operators of the hybrid GA–DSATUR algorithm. In the Develop stage, the system was implemented in CodeIgniter 3 using PHP and MySQL [28]; testing and debugging were carried out iteratively through black-box functional testing of each module and automatic verification that every generated timetable satisfied all hard constraints. The Disseminate stage concerned the deployment of the software in the production environment of the VCD Study Program.

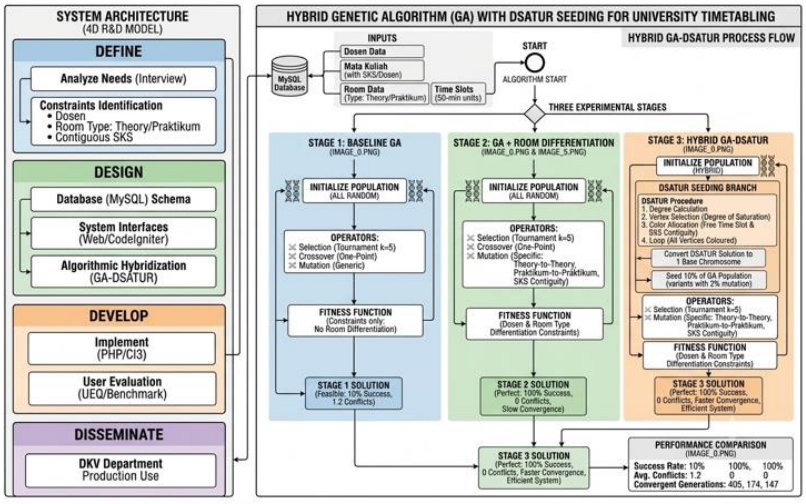


Figure 1. System architecture and workflow of the hybrid GA-DSATUR algorithm

Figure 1 consists of two main parts: the system architecture according to the 4D R&D model (Define, Design, Develop, and Disseminate) is shown on the left, while the workflow of the hybrid GA–DSATUR algorithm is shown on the right. The algorithm retrieves lecturer, course, room, and time-slot data from the MySQL database and processes them in three stages: Stage 1 (basic GA baseline without room-type differentiation), Stage 2 (GA with differentiated theory and practicum rooms), and Stage 3 (hybrid GA–DSATUR, in which DSATUR solutions seed 10% of the initial GA population). The evolutionary process, consisting of tournament selection, crossover, room- and credit-aware mutation, and fitness evaluation, is followed by a performance evaluation in which Stage 3 achieves a 100% success rate, zero conflicts, and the lowest mean convergence generation of the three stages.

Problem Representation

Graph Model for the DSATUR Algorithm

In the graph-coloring model, every section of the class is considered a vertex. We draw an edge from one class section to another if both are possibly incompatible. In our case, incompatible class sections are those

taught by the same lecturer; therefore, two sections that cannot be scheduled in the same time slot are connected. A colored node (a color to put it differently) symbolizes the time at which a section is scheduled. The DSATUR method assigns to each vertex a color different from the colors of its neighboring nodes, but in a way that the allocated time slots cover the entire credit hours without time gaps and are also completed during the same day.

Chromosome Representation for the Genetic Algorithm

Each chromosome is nothing more than a complete timetable solution. A single gene refers to one class section and carries these features: id_kelas, id_mata_kuliah, id_dosen, id_ruangan, id_waktu_start, and sks. The chromosome size is the same as the number of scheduled class sections. It is therefore 87 genes.

DSATUR Algorithm for Initial Population Seeding

The DSATUR algorithm [22] produces a feasible initial solution that is further refined by the GA. The procedure consists of four steps. The first step is the calculation of the saturation degree: the saturation degree of an uncolored vertex v is the number of distinct colors (time slots) already assigned to its neighbors, so a higher saturation degree indicates that v has fewer feasible options left. The second step is vertex selection: the uncolored vertex with the highest saturation degree is selected, and ties are broken by choosing the vertex with the highest degree, i.e., the largest number of neighbors. The third step is color assignment: the selected vertex receives the first available time slot that (a) is not used by any already colored neighbor and (b) provides enough consecutive slots on the same day to cover the credit value of the course. Finally, steps one to three are repeated until all vertices are colored.

The complete DSATUR solution is then converted into a base chromosome. This base chromosome is duplicated and subjected to light mutation with a probability of approximately 2% to create variations. These DSATUR-based variants make up 10% of the initial GA population (5 of 50 chromosomes), while the remaining 90% (45 chromosomes) are generated randomly, as in a standard GA. The seeding proportion of 10% was chosen to inject good building blocks while keeping most individuals random to preserve diversity, in line with reviews of initialization strategies warning that populations dominated by seeded individuals tend to converge prematurely [16]. A sensitivity analysis of the seeding proportion was not performed and is acknowledged as a limitation.

Genetic Algorithm Design

Parameter Settings

The GA parameters were set through preliminary tuning. A population size of 50 chromosomes and 500 generations were chosen because their product, 25,000 fitness evaluations, was considered sufficient to explore the search space of 87 class sections. A crossover rate of 0.8 is in line with the value most often used in the literature, whereas a mutation rate of 0.05 was selected to avoid premature convergence while preserving the structure of the solutions generated so far. Experiments with more aggressive parameters (a population of 100, 1,000 generations, and a mutation rate of 0.1) led to a marked decrease in performance: the mean fitness dropped by approximately 11.5%, from 87 to 77, with 8–12 conflicting sections per run. This degradation occurred because the high mutation rate disrupted good building blocks faster than selection could preserve them [14]. The final parameter settings were: population 50, 500 generations, crossover rate 0.8, mutation rate 0.05, tournament selection with k = 5, and elitism of one chromosome.

Fitness Function

The fitness function assesses how well a chromosome satisfies the hard constraints and is computed as Fitness = N − (V_L + V_R), where N is the number of class sections (87), V_L is the number of lecturer violations, and V_R is the number of room violations. A lecturer violation is counted for each pair of class sections taught by the same lecturer whose time slots overlap, and a room violation is counted for each pair of class sections assigned to the same room in overlapping time slots; each violation therefore decreases the fitness by 1. In this study, a conflict refers to any such violation. The theoretical maximum fitness equals the number of class sections, 87, which indicates a solution without any hard-constraint violation.

Genetic Operators

The tournament selection method was applied in the selection step with the tournament size set to k = 5. This particular selection strategy was adopted since it offers moderate selection pressure so that high-quality parents are preferentially picked, yet the population diversity is not lost due to the fact that very aggressive parents are not selected.

One-point crossover was used with a high probability of 0.8. At random, one position is chosen. From that point onward, the segments of genes are swapped with the corresponding parent, and two new offspring are generated.

Mutation was applied with a probability of 0.05 per gene using two room- and credit-aware operators: (a) moving a class section to another start slot that can accommodate its full credit load on the same day, and (b) reassigning the section to another room, restricted to rooms of the permitted type from Stage 2 onward. Both operators keep every mutated gene consistent with the contiguity constraint.

Elitism was applied by copying the single chromosome with the best fitness value in each generation unchanged into the next generation. This ensures that the best solution found so far is never lost during the evolutionary process.

Test Data

The test data were taken from the timetable of the odd semester of the 2025/2026 academic year (July–December 2025) of the VCD Study Program, Universitas Negeri Padang. They represent the total workload of one full semester: 87 class sections, 26 courses, 16 lecturers, 12 theory classrooms, and 6 practicum studios. Teaching hours were allocated over 5 days, from Monday to Friday, with 10 time slots per day, giving a total of 50 time slots per week. One 50-minute time slot corresponds to one credit unit, so a 3-credit course requires 3 consecutive time slots on the same day.

Evaluation Method

To examine the impact of each algorithmic change separately, the evaluation was carried out in three experimental stages, as explained below : a) Stage 1, basic GA (baseline): the GA had no room-type separation, so theory rooms and practicum rooms were treated identically. This stage represents the initial condition before any improvement was applied; b) Stage 2, GA with room-type separation: rooms were separated according to their type, so theory courses could only be scheduled in theory rooms and practicum courses only in practicum studios. The initial population was generated entirely at random, without DSATUR seeding; and c) Stage 3, hybrid GA–DSATUR with room-type separation: the GA was run with room-type separation, and its initial population was seeded with DSATUR solutions. Ten percent of the initial population consisted of DSATUR-based solution variants, and the remaining 90% was generated randomly.

Each stage was executed in ten independent runs, and every run used the full budget of 500 generations, so that computation times are comparable across runs. Reporting multiple independent runs is common practice for stochastic timetabling algorithms [2][31]; because ten runs limit statistical power, effect sizes are reported alongside p-values. For each run, the following data were recorded: (a) the convergence generation, defined as the first generation at which the best fitness of that run was reached; (b) the final best fitness; (c) the number of remaining conflicts; and (d) the elapsed computation time in seconds. A run was considered successful when it reached a fitness of 87 with zero conflicts. For each metric, the mean, standard deviation, minimum, maximum, and 95% confidence interval (CI) were computed. Success rates were compared using Fisher's exact test, while convergence generation and computation time were compared using Welch's t-test and confirmed with the Mann–Whitney U test, at a significance level of $\alpha = 0.05$; effect sizes were expressed as Cohen's d. All runs were executed on the same server configuration.

Results and Discussions

System Development Results

The web-based course timetabling system was successfully developed according to its design. The administrator interface provides master data management (courses, lecturers, rooms, etc.), GA parameter settings, GA execution, timetable optimization, and an interactive timetable view. Each row of the timetable shows the course, day, start time, end time (calculated from the credit load), room, and lecturer, and the timetable can be filtered by lecturer or room. Figures 2–5 document the implemented system, whereas the quantitative evaluation of the algorithm is reported in Tables 1–5.

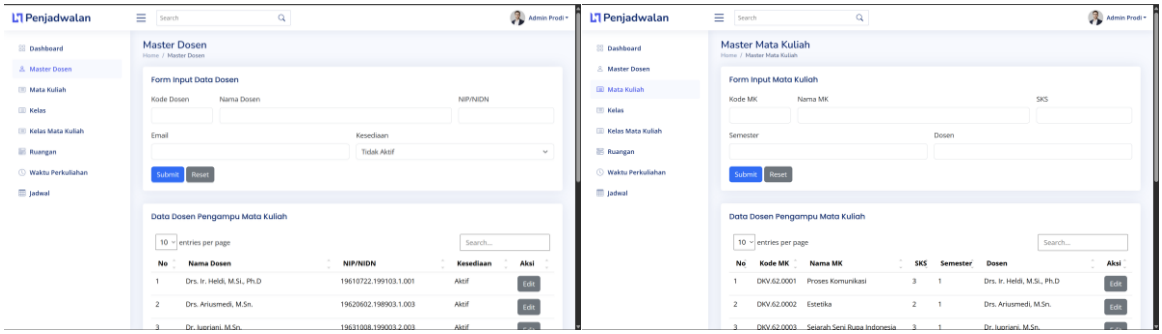


Figure 2. Lecturer Master Form and Course Form

Through the Lecturer Master Form interface the administrator may add, edit and delete a lecturer data who have teaching assignment in VCD Study Program. Besides, Course Form allows the administrator not only add, edit and delete courses, but also assigning the course to existing lecturer who have been selected earlier.

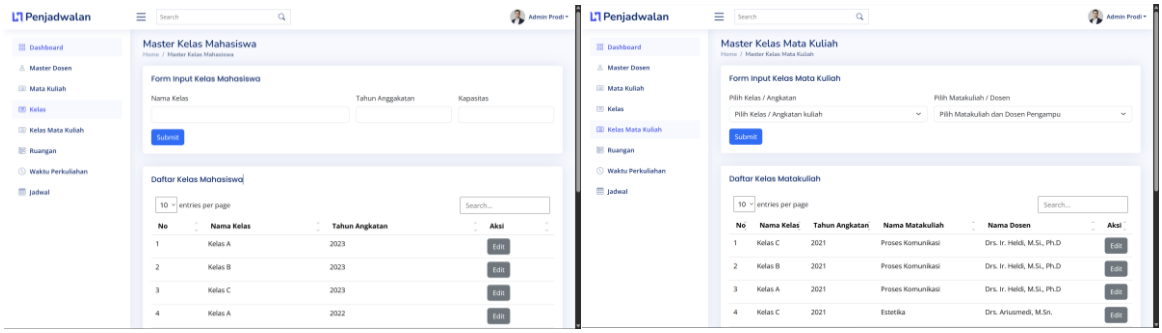


Figure 3. Class Form and Class–Course Form

Class Form allows the user to read, create, update, and delete data of classes. A lecturer is assigned to each class via a lecturer and class combination record in Class-Course Form.

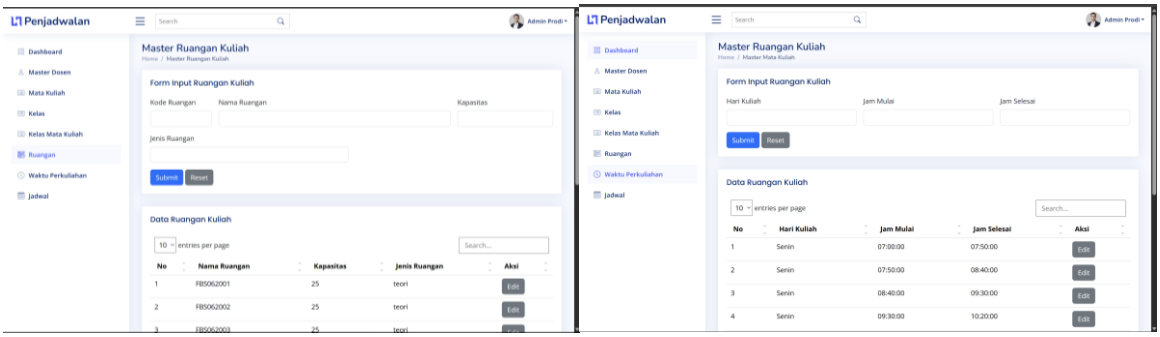


Figure 4. Room Form and Time Slot Form

The Room Form and the Time Slot Form are the means through which one manipulates room and teaching-time data such as CRUD operations. They also include a type of room, theory or practicum, that is considered when one schedules the timetable.

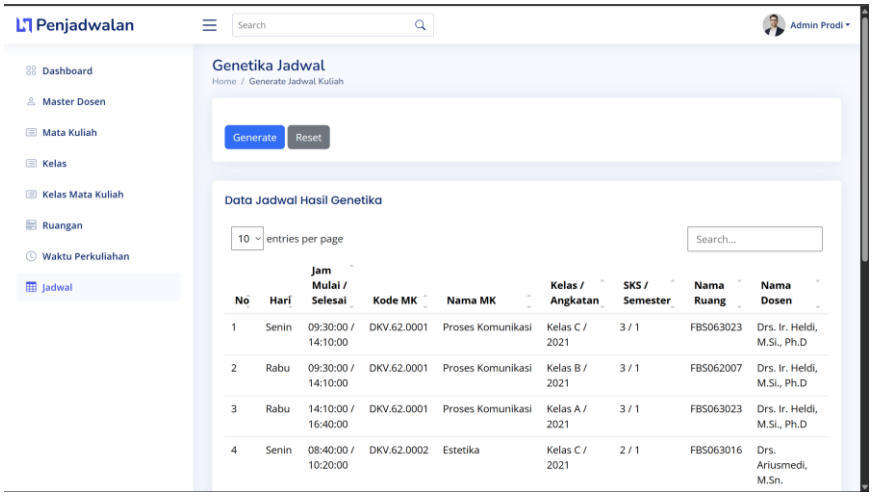


Figure 5. Timetable generation page

Figure 5 presents the timetable generation page. On this page, timetable generation experiments were done in three sessions, with ten executions per session. The created timetables are shown on this page.

Stage 1: Basic GA without Room-Type Separation (Baseline)
The initial stage establishes the baseline used when evaluating algorithmic performance prior to any change or enhancement related to constraint modeling. Table 1 shows a comparison of outcomes of the ten individual trials conducted in Stage 1.

Table 1. Performance of the basic GA without room-type separation (10 runs)

Run	Convergence Generation	Fitness	Remaining Conflicts	Time (s)
1	351	**87**	**0**	53.63
2	465	85	2	47.70
3	454	86	1	47.71
4	475	86	1	46.69
5	456	86	1	47.25
6	442	86	1	47.17
7	339	86	1	48.00
8	331	86	1	47.58
9	408	86	1	46.69
10	338	84	3	47.18
Mean	**405.9**	**85.8**	**1.2**	**47.96**
Std. Dev.	59.73	0.79	0.79	2.04
Min.	331	84	0	46.69
Max.	475	87	3	53.63

Table 1 shows that, without room-type differentiation, only 1 of 10 runs (10%) found a perfect solution, i.e., a fitness of exactly 87 without any conflict. The mean fitness was only 85.8 (95% CI 85.24–86.36), below the theoretical maximum of 87, with an average of 1.2 conflicts per timetable. Most of these conflicts were room-related, because the algorithm had no information on the required room type of each course. Consequently, practicum rooms were frequently allocated to theory courses and vice versa, which sometimes left no suitable room or time slot for some courses.

Stage 2: GA with Room-Type Separation
A further enhancement to the system was realized in Stage 2 when a rule was added that theory courses must be assigned only to theory rooms and practicum courses only to practicum rooms. The outcomes of the ten test runs are set out below in Table 2.

Table 2. Performance of the GA with room-type separation (10 runs)

Run	Convergence Generation	Fitness	Remaining Conflicts	Time (s)
1	95	87	0	72.90
2	118	87	0	67.44
3	199	87	0	83.84
4	290	87	0	68.75
5	93	87	0	68.79
6	160	87	0	65.49
7	154	87	0	66.15
8	242	87	0	68.61
9	149	87	0	70.55
10	244	87	0	79.10
Mean	**174.4**	**87**	**0**	**71.16**
Std. Dev.	67.30	0	0	5.93
Min.	93	87	0	65.49
Max.	290	87	0	83.84

Table 2 shows a substantial improvement: all runs (100%) were solved perfectly, with a fitness of 87 and zero conflicts, and the increase in the success rate from 10% to 100% was statistically significant (Fisher's exact test, $p < 0.001$). Room-type separation therefore proved empirically necessary to satisfy all hard constraints. By eliminating invalid room allocations, it considerably reduces the search space that the GA must explore, so that the algorithm consistently produces a feasible solution. The mean computation time increased from 47.96 s to 71.16 s (+48.4%) because additional queries are needed to validate the room type during initialization and mutation. Nonetheless, this extra time is well compensated by the essential gain in solution quality.

Stage 3: Hybrid GA-DSATUR with Room-Type Separation

In Stage 3, DSATUR-based seeding was added to the GA with room-type separation. Table 3 presents the results of the ten runs.

Table 3. Performance of the hybrid GA-DSATUR with room-type separation (10 runs)

Run	Convergence Generation	Fitness	Remaining Conflicts	Time (s)
1	228	87	0	52.07
2	91	87	0	57.05
3	174	87	0	63.15
4	70	87	0	86.79
5	112	87	0	79.40
6	93	87	0	71.74
7	155	87	0	53.02
8	93	87	0	66.95
9	202	87	0	62.91
10	255	87	0	49.11
Mean	**147.3**	**87**	**0**	**64.22**
Std. Dev.	65.12	0	0	12.29
Min.	70	87	0	49.11
Max.	255	87	0	86.79

All runs were again solved successfully. The mean convergence generation decreased to 147.3 (95% CI 100.7–193.9), and the mean computation time decreased to 64.22 s (95% CI 55.43–73.01). DSATUR seeding

therefore provides a better starting configuration that allows the GA, on average, to reach the optimal fitness in fewer generations, although the run-to-run variability remained high (SD = 65.12 generations).

Comprehensive Comparative Analysis

Along with this, Table 4 highlights comparison of performances from the three different stages of the experiments.

Table 4. Performance comparison of the three experimental stages

Metric	Stage 1 (Basic GA)	Stage 2 (GA + Room Separation)	Stage 3 (GA-DSATUR + Room Separation)
Mean Fitness	85.8	87	87
Mean Conflicts	1.2	0	0
Success Rate	10%	**100%**	**100%**
Mean Convergence Generation	405.9	174.4	147.3
Mean Computation Time (s)	47.96	71.16	64.22

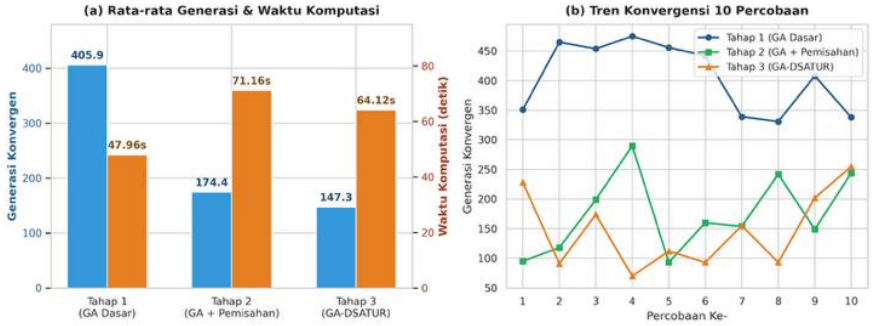


Figure 6. Comparison of mean convergence generation and computation time (a) and convergence trends over 10 runs (b)

Figure 6 presents the comparative evaluation of the three experimental stages. As shown in Figure 6(a), DSATUR seeding in Stage 3 reduced the mean convergence generation to 147.3, with a mean computation time of 64.22 s. The run-to-run convergence patterns in Figure 6(b) show that Stage 3 generally required fewer generations to reach the optimal solution than Stages 1 and 2.

From Stage 1 to Stage 2: A Leap in Quality (Constraint-Driven Improvement)

The biggest empirical evidence for our hypothesis that the main basis for a successful timetabling system is adequate constraint modeling and, in particular, a room-type separation comes from the jump of success rate from 10% to 100%. Without room-type separation the algorithm spends a large fraction of its fitness evaluations on combinations that should have been ruled out (i.e. putting practicum courses in theoretical rooms). This results not only in room conflicts but also has a domino-effect making the algorithm struggle for finding a globally feasible solution. One of the most important effects of restricting the search space to only domain-valid solutions is that the quality of the obtained solution greatly improves and the time increase from 47.96 s to 71.16 s is a fair cost for the added validation level.

From Stage 2 to Stage 3: A Leap in Efficiency (Heuristic-Driven Improvement)

After Stage 2 achieved consistently high-quality results, the next step was to improve efficiency. Seeding the initial population with DSATUR solutions instead of generating it purely at random reduced the mean convergence generation by 15.5% (147.3 vs 174.4) and the mean computation time by 9.8% (64.22 s vs 71.16 s). This reduction is attributed to the fact that DSATUR solutions have already systematically resolved lecturer conflicts from the first generation, so the remaining task of the GA is mainly room allocation, which is computationally lighter. However, Welch's t-test showed that neither reduction was statistically significant (convergence generation: $t(18.0) = 0.92$, $p = 0.372$, $d = 0.41$; computation time: $t(13.0) = 1.61$, $p = 0.132$, $d = 0.72$). The efficiency gain should therefore be interpreted as a consistent trend with a small-to-medium effect size rather than as a confirmed difference.

From Stage 1 to Stage 3: Holistic Improvement

Compared with the baseline, the final Stage 3 system shows a marked improvement: the success rate increased from 10% to 100%, and the mean convergence generation decreased by 63.7%, from 405.9 to 147.3 ($t(17.9) = 9.26$, $p < 0.001$, $d = 4.14$). The mean computation time in Stage 3 was 33.9% longer than in Stage 1 (64.22 s vs 47.96 s; $t(9.5) = -4.13$, $p = 0.002$), but this cost is clearly outweighed by the improvement in solution quality from a mean of 1.2 conflicts to zero conflicts. The improvement results mainly from integrating domain knowledge (room-type separation) with an algorithmic enhancement (DSATUR seeding).

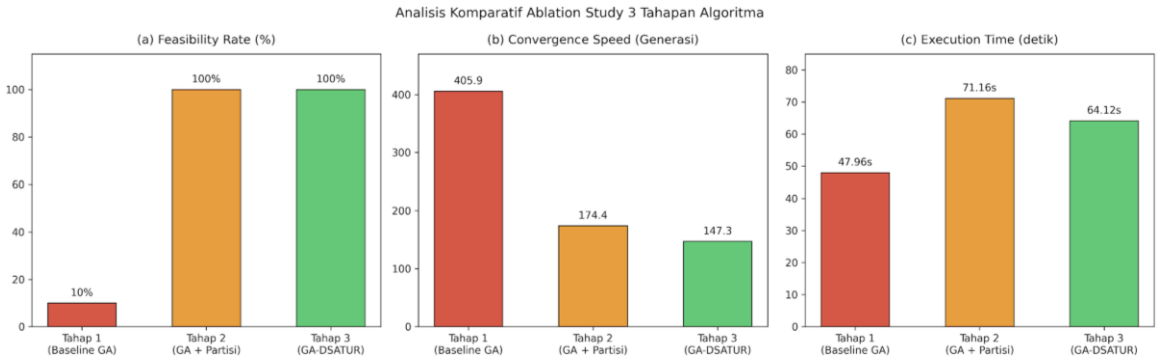


Figure 7. Comparative analysis of the three-stage ablation study

The results of the three-stage ablation study in Figure 7 illustrate the contribution of domain-based room partitioning and DSATUR-based population seeding to evolutionary search performance. Domain-based room partitioning in Stage 2 increased the feasibility rate from 10% to 100% (Figure 7a) and reduced the mean convergence generation from 405.9 to 174.4 (Figure 7b), although the execution time rose from 47.96 s to 71.16 s because of the added domain-validation logic. DSATUR population seeding in Stage 3 then reduced the mean convergence generation to 147.3 (63.7% lower than the baseline) and shortened the mean running time to 64.22 s. These experiments show that domain-based room partitioning is necessary to ensure solution validity (satisfying hard constraints), whereas DSATUR seeding mainly accelerates convergence and lowers computational overhead.

Table 5 summarizes the inferential tests between stages. Room-type separation (Stage 1 vs Stage 2) produced statistically significant differences with very large effect sizes, whereas the additional effect of DSATUR seeding (Stage 2 vs Stage 3) was not significant with ten runs, with small-to-medium effect sizes.

Table 5. Statistical comparison between experimental stages (10 runs per stage)

Comparison	Metric	Test (statistic, p-value)	Effect size
Stage 1 vs Stage 2	Success rate (10% vs 100%)	Fisher's exact, $p < 0.001$	–
Stage 1 vs Stage 2	Convergence generation	Welch $t(17.7) = 8.14$, $p < 0.001$	$d = 3.64$
Stage 1 vs Stage 2	Computation time	Welch $t(11.1) = -11.71$, $p < 0.001$	$d = 5.24$
Stage 2 vs Stage 3	Convergence generation	Welch $t(18.0) = 0.92$, $p = 0.372$	$d = 0.41$
Stage 2 vs Stage 3	Computation time	Welch $t(13.0) = 1.61$, $p = 0.132$	$d = 0.72$
Stage 1 vs Stage 3	Convergence generation	Welch $t(17.9) = 9.26$, $p < 0.001$	$d = 4.14$
Stage 1 vs Stage 3	Computation time	Welch $t(9.5) = -4.13$, $p = 0.002$	$d = 1.85$

The Critical Role of Room-Type Separation

This impressive increase in success rate from 10% to 100% is quite simply a case of constraint programming as well as search space reduction theory that can be applied and understood to explain the finding that room-type separation has a big success rate. In case of absence of room-type separation the quotient of feasible solutions relative to the whole search space is minimal. A genetic algorithm based totally on chance selection is highly unlikely to choose a feasible region as its starting point and thus frequently does not succeed in finding solutions which fit all the constraints even after 500 generations. Room-type separation works by the fact that the infeasible room combinations are removed and hence it is as if the proportion of the search space which is feasible is being enhanced and consequently the chances of the GA being successful in the search and the holding on to the solution will be higher.

Acceleration Mechanism of DSATUR Seeding

DSATUR is good for seeding because of its power to deterministically and systematically resolve lecturer conflicts. Through its selection of the ones having higher degrees of saturation, the method guarantees that the sections most subject to conflicts of lectures will be scheduled first. This approach not only makes a time slot "safe" for those entities that have limited options before handling those that are not that restricted but also reduces subsequent lecturer–time conflicts. Once DSATUR-based solutions are seeded in the GA population, the evolutionary algorithm (EA) is no longer required to search basic timetable structure from scratch. Instead, the GA can work immediately for the optimization of room allocation and making small changes to time slots.

This observation is consistent with research on hybrid metaheuristics [16][17][29], which shows that combining heuristics that generate initial solutions with evolutionary algorithms often performs better than using either method alone. A further result of our study is the numerical evidence that seeding the GA population with graph-coloring solutions lowers the mean number of generations needed to converge even when the GA alone is already able to find perfect solutions, although with ten runs this reduction was not statistically significant.

Parameter Sensitivity Analysis

We ran the same tests with more aggressive parameters (a population of 100, 1,000 generations, and a mutation rate of 0.1) to examine the influence of the GA parameters on the output. Performance degraded markedly under these settings (mean fitness of 77, with 8–12 remaining conflicts per run), which indicates that more computation time and resources do not necessarily lead to better results. When the mutation rate is too high (0.1), the GA behaves almost like a random search because good gene structures are broken faster than selection can preserve them. This finding supports the main principle of GA design, namely balancing exploration (through mutation) and exploitation (through selection and crossover) [14].

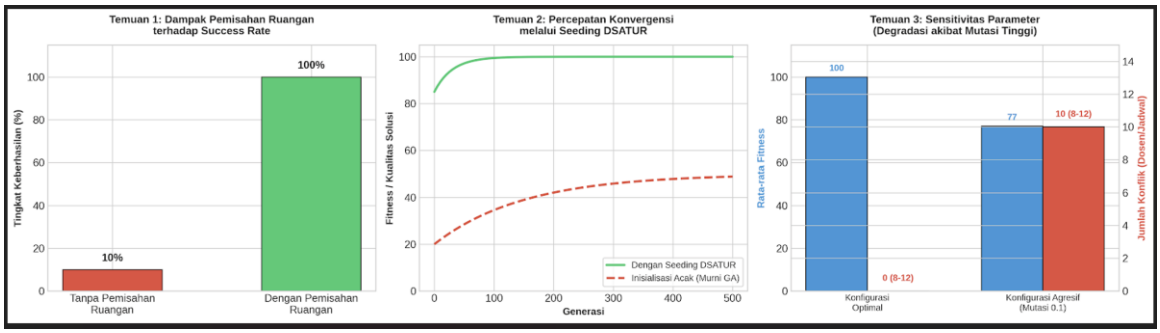


Figure 8. Summary of key findings

The results indicate that the combination of room-type separation, which reduces the number of possible combinations, and DSATUR seeding, which provides informed starting points, underlies the improvement in the success rate from 10% to 100% and the faster convergence. A pure GA without domain constraints is clearly inadequate for a problem as constrained as the University Course Timetabling Problem (UCTP), whereas the hybrid configuration achieved the best overall performance among the three configurations tested.

Comparison with Previous Studies and Theoretical Implications

The results are consistent with surveys showing that the correct modeling of hard constraints largely determines whether metaheuristics can reach feasible timetables [1][2][4][5][8][34]. Surveys of exam and school timetabling report the same pattern, in which constructive graph-coloring heuristics are used to obtain feasible starting solutions that are subsequently improved by metaheuristics [12][13][33][35]. The increase from 10% to 100% in the success rate between Stage 1 and Stage 2 provides a quantified confirmation of this principle for a studio-based program, a setting in which the room-type dimension has received little attention compared with curriculum-based or post-enrolment benchmarks [2][6][7].

Regarding hybridization, earlier GA-based timetabling studies improved performance through parallel GA with local search [17], dynamic chromosomes [15], robustness-oriented hybrid GAs [30], memetic or hyper-heuristic frameworks [23][27], and graph-coloring-based GA analysis [29], while GA case studies at individual institutions [10][11] reported feasible timetables without isolating the effect of each component. Graph-coloring methods alone, such as DSATUR or Welsh–Powell [18][20][21], produce feasible timetables quickly but do not optimize them further. Our results agree with reports that heuristic initialization improves

population-based metaheuristics [16], but they also indicate that, once the domain constraints are correctly modeled, the additional benefit of seeding is moderate (15.5% fewer generations, d = 0.41) and not statistically significant with ten runs. Exact approaches such as mixed-integer programming [19][32] can guarantee optimality but require specialized solvers, whereas the proposed GA–DSATUR system reached conflict-free timetables in about one minute using a standard PHP–MySQL stack, similar to other web-based GA implementations [24][26].

Theoretically, these findings support the view of timetabling as a constraint satisfaction problem in which reducing the infeasible portion of the search space has a larger effect on evolutionary search than improving the starting point [31][34]. Practically, they suggest that developers of timetabling systems for programs with specialized facilities should prioritize accurate modeling of facility types before investing in more sophisticated initialization or hybridization strategies.

Limitations and Future Research

Even though the method perfectly satisfied hard constraints and the results showed a significant gain, still there are several shortcomings that need to be addressed as a topic for further research.

First, "soft constraints" like instructors' preferred teaching times, an even spread of teaching loads throughout the week, and preventing student idle "gaps" have not been taken into count in the "fitness" function yet. Including such constraints definitely raises the level of acceptance as well as actual usefulness of timetables that come out of the model.

Secondly, cross-program timetabling is still a hypothetical approach since it has not yet been field-tested, and may have varied implementation conditions.

Thirdly, the experiment used a single dataset from one study program and ten runs per stage, which limits statistical power and external validity. Institutions with more class sections, rooms shared across programs, student-group constraints, or different credit-hour structures may require additional constraints and parameter re-tuning, and the 10% seeding proportion was not subjected to a sensitivity analysis.

Given these limitations, we recommend that further studies (1) include soft constraints and an adaptive penalty in the fitness model; (2) evaluate scalability and general applicability by running the hybrid algorithm on datasets from different semesters and study programs, as well as on public benchmarks such as ITC2007 and ITC2019 [2]; and (3) increase the number of runs and test several seeding proportions to strengthen the statistical evidence on the efficiency effect of DSATUR seeding.

Conclusions

This paper reported the development of a web-based automated course timetabling system based on a hybrid GA–DSATUR algorithm. The system implements two important constraints: (a) credit-based contiguity, which assigns consecutive time slots on the same day according to the course credit load; and (b) room-type separation, which allocates theory courses to theory rooms and practicum courses to studios or laboratories. The three-stage experiment yielded two main findings. First, room-type separation is a necessary precondition for feasibility: it increased the success rate from 10% to 100% (Fisher's exact test, $p < 0.001$) and reduced the mean convergence generation from 405.9 to 174.4 ($p < 0.001$, $d = 3.64$). Second, DSATUR seeding reduced the mean convergence generation by 15.5% and the mean computation time by 9.8% (64.22 s vs 71.16 s); these reductions were consistent in direction but not statistically significant with ten runs ($p = 0.372$ and $p = 0.132$). The final system produced conflict-free timetables in every run, reducing a manual process that previously took several days to about one minute of computation. The contribution of this study is an ablation-based quantification showing that, for studio-based programs, domain constraint modeling contributes far more to solution feasibility than heuristic seeding, which mainly offers a moderate efficiency gain. The framework is likely to transfer to institutions with similar room-type structures and lecturer-centered conflicts; however, its generalizability is limited by the single-program dataset, and the approach may perform less well when student-group conflicts, shared facilities, or soft constraints dominate. Future work should validate the framework on other programs and public benchmarks, increase the number of runs, and incorporate soft constraints.

Acknowledgments

We thank the Institute for Research and Community Service (LPPM), Universitas Negeri Padang, for funding this work under contract number 1616/UN35-15/LT/2025.

References

S. Abdipoor, R. Yaakob, S. L. Goh, and S. Abdullah, "Meta-heuristic approaches for the University Course Timetabling Problem," Intelligent Systems with Applications, vol. 19, p. 200253, Sep. 2023, doi: 10.1016/J.ISWA.2023.200253.

S. Ceschia, L. Di Gaspero, and A. Schaerf, "Educational timetabling: Problems, benchmarks, and state-of-the-art results," *European Journal of Operational Research*, vol. 308, no. 1, pp. 1–18, Jul. 2023, doi: 10.1016/j.ejor.2022.07.011.

Y. Xiao, X. L. Li, L. Jiang, P. Wang, K. Wang, and N. Luo, "Towards Dynamic University Course Timetabling Problem: An Automated Approach Augmented via Reinforcement Learning," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 520–529, 2024, doi: 10.1109/ICDM59182.2024.00059.

M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar, and G. Kendall, "A survey of university course timetabling problem: Perspectives, trends and opportunities," *IEEE Access*, vol. 9, pp. 106515–106529, 2021, doi: 10.1109/ACCESS.2021.3100613.

H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, "A Review of Optimization Algorithms for University Timetable Scheduling," *Engineering, Technology & Applied Science Research*, vol. 10, no. 6, pp. 6410–6417, Dec. 2020, doi: 10.48084/ETASR.3832.

M. Davison, A. Kheiri, and K. G. Zografos, "Modelling and solving the university course timetabling problem with hybrid teaching considerations," *Journal of Scheduling 2024 28:2*, vol. 28, no. 2, pp. 195–215, Oct. 2024, doi: 10.1007/S10951-024-00817-W.

G. Colajanni and P. Daniele, "A new model for curriculum-based university course timetabling," *Optimization Letters 2020 15:5*, vol. 15, no. 5, pp. 1601–1616, May 2020, doi: 10.1007/S11590-020-01588-X.

H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem," *Comput. Ind. Eng.*, vol. 86, pp. 43–59, Aug. 2015, doi: 10.1016/J.CIE.2014.11.010.

M. Mokhtari, M. Vaziri Sarashk, M. Asadpour, N. Saeidi, and O. Boyer, "Developing a Model for the University Course Timetabling Problem: A Case Study," *Complexity*, vol. 2021, no. 1, p. 9940866, Jan. 2021, doi: 10.1155/2021/9940866.

A. M. Al-Fitouri, A. Agaal, M. M. Ali, and H. A. Bugrin, "A Genetic Algorithm Approach for University Course Timetabling: A Case Study at the Faculty of IT, Sebha University," *Sebha University Conference Proceedings*, vol. 4, no. 3, pp. 173–179, Oct. 2025, doi: 10.51984/sucp.v4i3.4003.

I. A. Abduljabbar, S. M. Abdullah, I. A. Abduljabbar, and S. M. Abdullah, "An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem," *Baghdad Science Journal*, vol. 19, no. 2, p. 11, Apr. 2022, doi: 10.21123/bsj.2022.19.2.0399.

A. Ghaffar, I. U. Din, A. Tariq, and M. H. Zafar, "Hybridization and artificial intelligence in optimizing university examination timetabling problem: A systematic review," *Review of Education*, vol. 13, no. 2, p. e70071, Aug. 2025, doi: 10.1002/REV3.70071.

E. Sing et al., "A Survey of Solution Methodologies for Exam Timetabling Problems," *IEEE Access*, vol. 12, pp. 41479–41498, 2024, doi: 10.1109/ACCESS.2024.3378054.

S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications 2020 80:5*, vol. 80, no. 5, pp. 8091–8126, Oct. 2020, doi: 10.1007/S11042-020-10139-6.

G. Alnowaini and A. A. Aljomai, "Genetic Algorithm for Solving University Course Timetabling Problem Using Dynamic Chromosomes," *2021 International Conference of Technology, Science and Administration, ICTSA 2021*, Mar. 2021, doi: 10.1109/ICTSA52017.2021.9406539.

J. O. Agushaka and A. E. Ezugwu, "Initialisation Approaches for Population-Based Metaheuristic Algorithms: A Comprehensive Review," *Applied Sciences 2022, Vol. 12, Page 896*, vol. 12, no. 2, p. 896, Jan. 2022, doi: 10.3390/APP12020896.

A. Rezaeipanah, S. S. Matoori, and G. Ahmadi, "A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search," *Applied Intelligence 2020 51:1*, vol. 51, no. 1, pp. 467–492, Aug. 2020, doi: 10.1007/S10489-020-01833-X.

S. Biswas, S. A. Nusrat, N. Sharmin, and M. Rahman, "Graph Coloring in University Timetable Scheduling," *International Journal of Intelligent Systems and Applications*, vol. 15, no. 3, pp. 16–32, Jun. 2023, doi: 10.5815/IJISA.2023.03.02.

K. Xiang, X. Hu, M. Yu, and X. Wang, "Exact and heuristic methods for a university course scheduling problem," *Expert Syst. Appl.*, vol. 248, p. 123383, Aug. 2024, doi: 10.1016/J.ESWA.2024.123383.

M. Farhan, "Pewarnaan Graf dalam Penjadwalan Mata Pelajaran SMP Menggunakan Algoritma DSATUR," 2025, Accessed: Aug. 13, 2026. [Online]. Available: http://repository.ipb.ac.id/handle/123456789/160936

R. N. Amalia and P. Affandi, "PENERAPAN PEWARNAAN GRAF UNTUK OPTIMALISASI PENJADWALAN KULIAH DI PROGRAM STUDI MATEMATIKA: Algoritma Welch-Powell," *Equiva Journal*, vol. 3, no. 1, pp. 34–42, Jul. 2025, Accessed: Aug. 13, 2026. [Online]. Available: https://journal.itk.ac.id/index.php/equiva/article/view/1240

D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, Apr. 1979, doi: 10.1145/359094.359101.

L. Cheng, Q. Tang, L. Zhang, and Z. Zhang, "Multi-objective Q-learning-based hyper-heuristic with Bi-criteria selection for energy-aware mixed shop scheduling," *Swarm Evol. Comput.*, vol. 69, p. 100985, Mar. 2022, doi: 10.1016/J.SWEVO.2021.100985.

M. C. Yustina, I. N. Ichsan, and G. M. Suranegara, "Implementasi Algoritma Genetika Proses Mutasi Differential Evolution Pada Sistem Penjadwalan Mata Pelajaran," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 5, no. 1, pp. 116–130, Aug. 2024, doi: 10.30865/KLIK.V5I1.2109.

Y. Indarta, F. Ranuharja, S. Sukardi, A. D. S. Samala, and I. P. Dewi, "Web Application Development for Syari Online Shop Using 4D Model Development," *Sinkron : jurnal dan penelitian teknik informatika*, vol. 6, no. 2, pp. 720–728, May 2022, doi: 10.33395/SINKRON.V7I2.11376.

D. Romaguera, J. Plender-Nabas, J. Matias, and L. Austero, "Development of a Web-based Course Timetabling System based on an Enhanced Genetic Algorithm," *Procedia Comput. Sci.*, vol. 234, pp. 1714–1721, Jan. 2024, doi: 10.1016/J.PROCS.2024.03.177.

A. Ghaffar, M. U. Sattar, M. Munir, and Z. Qureshi, "Multi-objective fuzzy-based adaptive memetic algorithm with hyper-heuristics to solve university course timetabling problem," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 9, no. 4, pp. e4–e4, Dec. 2021, doi: 10.4108/EAI.16-12-2021.172435.

M. Herman, "Designing a Web-Based Thesis Title Submission System Using the Codeigniter 3 Framework," *Knowbase : International Journal of Knowledge in Database*, vol. 1, no. 1, pp. 44–50, Jun. 2021, doi: 10.30983/IJOKID.V1I1.4960.

M. Assi, B. Halawi, and R. A. Haraty, "Genetic algorithm analysis using the graph coloring method for solving the university timetable problem," *Procedia Computer Science*, vol. 126, pp. 899–906, 2018, doi: 10.1016/j.procs.2018.08.024.

C. Akkan and A. Gülcü, "A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem," *Computers & Operations Research*, vol. 90, pp. 22–32, Feb. 2018, doi: 10.1016/j.cor.2017.09.007.

S. L. Goh, G. Kendall, and N. R. Sabar, "Improved local search approaches to solve the post enrolment course timetabling problem," *European Journal of Operational Research*, vol. 261, no. 1, pp. 17–29, Aug. 2017, doi: 10.1016/j.ejor.2017.01.040.

E. Rappos, E. Thiémard, S. Robert, and J.-F. Hêche, "A mixed-integer programming approach for solving university course timetabling problems," *Journal of Scheduling*, vol. 25, no. 4, pp. 391–404, Aug. 2022, doi: 10.1007/s10951-021-00715-5.

J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems," *Expert Systems with Applications*, vol. 165, p. 113943, Mar. 2021, doi: 10.1016/j.eswa.2020.113943.

R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectrum*, vol. 30, no. 1, pp. 167–190, 2008, doi: 10.1007/s00291-007-0097-0.

R. Qu, E. K. Burke, and B. McCollum, "Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems," *European Journal of Operational Research*, vol. 198, no. 2, pp. 392–404, 2009, doi: 10.1016/j.ejor.2008.10.001.