

Automatic University Course Scheduling Using A Multi-Constraint Genetic Algorithm For Room and Time Optimization

Sapriansyah^{1*}, Henry Pratiwi², Ahmad Fahrijal Pukeng³

¹Department of Informatics Engineering, STMIK Widya Cipta Dharma, Indonesia.

²Department of Information Systems, STMIK Widya Cipta Dharma, Indonesia.

* Corresponding author:

Email: henypratiwi@wicida.ac.id

Abstract.

University course scheduling required simultaneous handling of resource conflicts, room capacity, and schedule quality. This study developed a Multi-Constraint Genetic Algorithm (MCGA) for automatic course scheduling using institutional data from STMIK Widya Cipta Dharma. The final dataset contained 141 optimized course offerings, 12 candidate rooms, five scheduling days, and six nominal time slots; room-capacity compliance was evaluable for 129 offerings (91.49%). A Standard Genetic Algorithm and MCGA were evaluated using identical feasibility domains and hard-conflict repair mechanisms. One-factor-at-a-time screening selected a population of 100, 500 generations, a crossover rate of 0.80, and a mutation rate of 0.02. Across 20 paired runs, both algorithms achieved zero lecturer, room, and cohort conflicts, with zero capacity excess among course offerings for which validated class-size data were available. MCGA increased mean room utilization from 0.6085 to 0.7800 and reduced mean cohort-day time gaps from 11.85 to zero. Paired Wilcoxon tests confirmed significant improvements in room utilization and time-gap reduction. The results showed that hierarchical feasibility handling combined with weighted post-feasibility objectives improved the quality of feasible university schedules without a material runtime penalty.

Keywords: University course timetabling, Multi-constraint optimization, Genetic Algorithm, Room utilization and Schedule compactness.

I. INTRODUCTION

University course timetabling is a combinatorial optimization problem in which courses must be assigned to limited rooms and time periods while satisfying lecturer, student, resource, and institutional constraints. Hard constraints determine schedule feasibility, whereas soft constraints distinguish the quality of feasible schedules, including room suitability, workload distribution, and idle-time reduction. Consequently, university timetabling remains an active research area involving exact, heuristic, metaheuristic, and hybrid optimization approaches [1], [2], [3], [4].

Previous studies have addressed university timetabling through mathematical programming, multi-criteria optimization, local search, hybrid methods, and Genetic Algorithm (GA)-based approaches [5], [6], [7], [8], [9], [10], [11]. GA and related metaheuristics are particularly useful for large scheduling spaces, although their performance depends on constraint representation, fitness design, improvement mechanisms, and parameter configuration [12], [13], [14], [15], [16].

In the Indonesian context, GA-based scheduling has incorporated institution-specific requirements such as lecturer availability, day-off requests, and room capacity [17], [18]. At STMIK Widya Cipta Dharma, Pongsumarre, Wahyuni, and Fahmi applied a Java-based Ant Colony Optimization approach to university course scheduling and reported conflict-free timetables across repeated runs while soft-constraint penalties continued to improve during optimization [19]. Parameter configuration is also important; previous Indonesian research showed that crossover probability can affect GA behavior and computation time [20].

Despite these developments, limited attention has been given to controlled GA comparisons that explicitly separate hard-feasibility handling from post-feasibility schedule-quality optimization. This study therefore investigates whether a weighted multi-constraint fitness can continue improving room utilization and cohort-day time compactness after hard conflicts have been eliminated when Standard GA and the proposed method use the same capacity-feasible room domain and hard-conflict repair mechanism.

This study proposes a Multi-Constraint Genetic Algorithm (MCGA) using hierarchical feasibility domination, a capacity-feasible room domain, and common hard-conflict repair. After feasibility is prioritized, MCGA optimizes two active soft objectives: room utilization and cohort-day time distribution.

Lecturer preference is inactive because validated preference data are unavailable. MCGA is evaluated against Standard GA using the same dataset, search domain, repair mechanism, parameter configuration, and paired random seeds. The study addresses three questions: (RQ1) how GA can be designed under multiple hard and soft constraints; (RQ2) how MCGA performs compared with Standard GA in feasibility, room utilization, time compactness, and computational time; and (RQ3) how key GA parameters influence solution quality.

II. METHODS

2.1. Study Data and Preprocessing

The study used the theoretical course timetable for the even semester of academic year 2025/2026 at STMIK Widya Cipta Dharma. Student counts were matched with the institutional final-examination schedule for room-capacity evaluation. The dataset contained 142 course offerings, of which 141 were optimized and one external offering was retained as fixed. The optimized dataset included 40 lecturers, three study programs, 12 candidate rooms, five scheduling days, and six nominal time slots. Validated class-size data were available for 129 of 141 optimized offerings (91.49%).

Table 1. Summary of the final scheduling dataset

Item	Value
Total source offerings	142
Offerings optimized by GA	141
Fixed/external offerings	1
Unique lecturers	40
Study programs	3
Candidate rooms	12
Candidate days	5 (Monday-Friday)
Nominal time slots	6
Validated class-size coverage	129/141 (91.49%)

Room capacity was represented by an empirical operational proxy rather than official inventory capacity. For each room, the proxy was the maximum observed examination occupancy or validated regular-class size previously assigned to that room. These values therefore represent operational lower bounds rather than official capacity certificates.

Table 2. Empirical room-capacity proxies used in the search domain

Room	Proxy	Room	Proxy	Room	Proxy
1	39	7	45	11	55
2	42	8	40	12	42
3/5	46	9	30	13	24
4/6	64	10	25	14	8

2.2. Solution Representation and Search Domain

Each chromosome represented one complete timetable. A gene corresponded to one course offering and contained only the decision variables day, time slot, and room; course identity, lecturer, cohort, and validated class size were fixed attributes. Direct and evolutionary encodings of course timetabling are common in GA-based scheduling, while related studies also incorporate student preferences and professor-assignment decisions [21], [22], [23]. The implemented representation was:

$$Gene_i = (day_i, slot_i, room_i), \quad X = [Gene_1, Gene_2, \dots, Gene_n].$$

The search space was restricted using institutional class-group patterns. Day-class groups P, PA, and PB were assigned to the first three nominal slots, while M, MA, and P/M groups were assigned to the last three slots. Unknown labels could use all slots. For offerings with validated class sizes, only rooms with sufficient empirical capacity were included in the feasible room domain:

$$AllowedRooms(i) = \{ r \mid C_r \geq S_i \}. \quad (1)$$

If class size was not validated, all candidate rooms remained available and capacity compliance for that offering was not claimed. This avoided fabricating unknown enrollment data.

2.3. Hard and Soft Constraints

The model used four hard constraints. HC1 prevented the same lecturer from receiving more than one offering in the same day-slot. HC2 prevented simultaneous use of the same room. HC3 prevented the same cohort key from receiving more than one offering in the same day-slot; this was a cohort-level constraint rather than an individual student-enrollment constraint. In the implementation, the cohort key represented an institutional class group defined from the study program, semester, and class-group designation, so that offerings belonging to the same academic cohort were prevented from occupying the same day-slot. HC4 evaluated room-capacity overflow only for offerings with validated class sizes. Constraint-based and evolutionary timetabling studies similarly distinguish hard feasibility from soft satisfaction [24], [25], [26].

$$HC4(X) = \sum_i \max(0, S_i - C_{(r_i)}). \quad (2)$$

The final MCGA used two active soft constraints. SC2 measured room-use efficiency. For every capacity-evaluable offering, utilization was capped at 1.0 because overflow was already treated as a hard violation:

$$U_i = \min(S_i / C_{(r_i)}, 1), \quad SC2 = \text{mean}(1 - U_i). \quad (3)$$

SC3 measured cohort-day compactness. For each cohort-day containing at least two scheduled classes, an internal gap was the number of unused slots between the first and last occupied slot. The normalized penalty was computed locally so that SC3 remained on a 0-1 scale comparable to SC2:

$$\text{gap}(g,d) = (\text{last_slot} - \text{first_slot} + 1) - \text{used_slots}(g,d), \quad (4)$$

$$SC3 = \text{mean}[\text{gap}(g,d) / \max(1, T - \text{used_slots}(g,d))]. \quad (5)$$

where T denotes the number of nominal time slots (T = 6).

Lecturer preference was originally considered as a possible soft constraint, but it was inactive in the final experiment because no validated preference or availability dataset was available.

2.4. Fitness, Feasibility Domination, and Hard-Conflict Repair

The Standard GA comparator used an equal raw penalty for lecturer, room, cohort, and capacity conflicts and did not include soft objectives.

Accordingly, the Standard GA penalty and fitness were defined as:

$$\text{Penalty_Standard} = HC1 + HC2 + HC3 + HC4, \quad (6)$$

$$\text{Fitness_Standard} = 1 / (1 + \text{Penalty_Standard}). \quad (7)$$

Specifically, Hard_normalized was computed as $(HC1 + HC2 + HC3)/n + HC4/\sum_{i \in V} S_i$, where n denotes the number of optimized course offerings and V denotes the subset of offerings with validated class-size data. A hard-dominance factor of 100 was then applied before the weighted soft penalty. Room utilization and time distribution were assigned equal weights of 0.50:

$$\text{Penalty_MCGA} = 100 \times \text{Hard_normalized} + 0.50 \times SC2 + 0.50 \times SC3, \quad (8)$$

$$\text{Fitness_MCGA} = 1 / (1 + \text{Penalty_MCGA}). \quad (9)$$

Raw fitness values were not compared between Standard GA and MCGA because the formulas used different scales. Instead, both algorithms were compared using common metrics such as hard conflicts, capacity excess, room utilization, time gaps, and execution time.

Both algorithms used the same hierarchical feasibility domination, ranking solutions first by HC1-HC3 conflicts, then by capacity excess, and finally by the algorithm-specific secondary criterion. The same hard-only conflict-aware repair was also applied to both algorithms. Candidate assignments were accepted only when they reduced the local hard-conflict score. During evolution, repair used probability 0.10, one repair step, and up to 30 candidates per repaired gene; the final best chromosome received up to 50 repair steps. The repair did not optimize the soft objectives [27], [28], [29], [30].

2.5. Genetic Operators and Final Parameters

The initial population was randomized and did not use the institutional baseline as a warm start. Parent selection used tournament selection with tournament size 3. Two-point crossover exchanged

contiguous blocks of course-offering genes. Mutation selected one decision field (day, time, or room) for a mutated gene and sampled a new value from the corresponding permitted domain. Elitism retained the two best individuals in each generation.

Table 3. Final GA and repair configuration

Parameter	Value	Parameter	Value
Population size	100	Tournament size	3
Generations	500	Elite size	2
Crossover rate	0.80	Soft weights	0.50 / 0.50
Mutation rate	0.02	Hard dominance	100
Repair probability	0.10	Repair candidate limit	30
Repair steps/evolution	1	Final repair max steps	50

2.6. Experimental Design and Statistical Analysis

RQ3 was evaluated using one-factor-at-a-time (OFAT) screening. The baseline used population 100, 500 generations, crossover 0.80, and mutation 0.05. Alternative levels were $P = 50/150$, $G = 250/750$, $CR = 0.70/0.90$, and $MR = 0.02/0.10$. Nine unique configurations were evaluated using seeds 11, 22, and 33 and ranked by feasibility, residual hard conflicts, soft quality, and execution time.

The soft-quality score used for parameter ranking was defined as:

$$\text{SoftQuality} = 0.50 \times (1 - \bar{U}) + 0.50 \times \text{SC3}, \quad (10)$$

where $\bar{U}$ denotes the mean room-utilization ratio and lower SoftQuality values indicate better post-feasibility schedule quality.

After the parameter configuration was frozen, The final comparison used 20 paired independent seeds for each algorithm, producing 40 runs. Paired results were evaluated using descriptive statistics, two-sided Wilcoxon signed-rank tests, and a 95% confidence interval for room-utilization differences. Seed 101 was used for representative convergence analysis. Execution time was measured in Python/Google Colab and interpreted as an environment-dependent indicator.

III. RESULT AND DISCUSSION

3.1. Parameter Screening

All nine screening configurations eliminated lecturer, room, and cohort conflicts, with no capacity excess in the validated subset. Post-feasibility quality nevertheless differed substantially. $MR = 0.02$ achieved the highest mean room utilization (0.7848) with zero time gaps and zero SC3 penalty, whereas $MR = 0.10$ produced utilization of 0.6020 and 12 mean gaps. Thus, among the tested levels, mutation rate showed the clearest influence on schedule quality, consistent with previous findings on GA parameter sensitivity [20].

Table 4. OFAT parameter-screening results

Rank	Configuration	Room util.	Mean gap	SC3	Time (s)
1	MR=0.02	0.7848	0.00	0.0000	158.7
2	G=750	0.6561	2.00	0.0125	246.9
3	CR=0.7	0.6534	3.33	0.0208	165.3
4	CR=0.9	0.6442	2.67	0.0161	165.4
5	BASE	0.6492	3.33	0.0211	165.6
6	P=50	0.6391	3.67	0.0239	82.0
7	P=150	0.6330	4.33	0.0271	249.9
8	G=250	0.6385	7.67	0.0461	83.2
9	MR=0.1	0.6020	12.00	0.0754	172.8

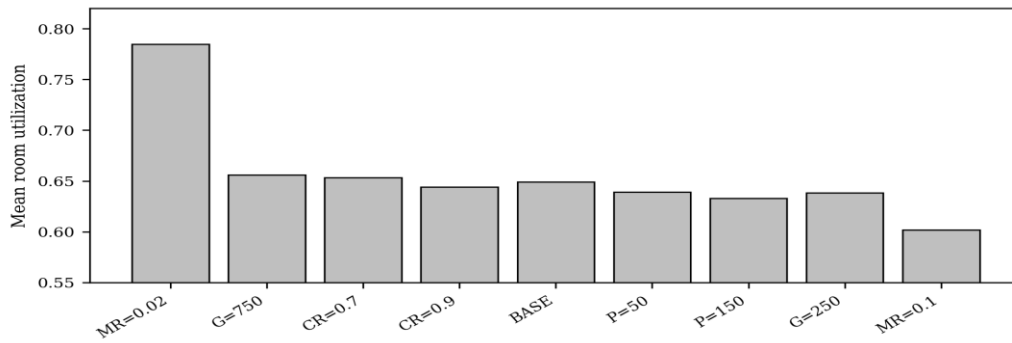


Fig. 1. Mean room utilization under the tested MCGA parameter configurations.

3.2. Final Feasibility and Schedule Quality

Both Standard GA and MCGA eliminated lecturer, room, and cohort conflicts across all 20 runs, with zero capacity excess in the validated class-size subset. This indicates that the common feasibility domain, domination strategy, and hard-only repair were sufficient for feasibility.

MCGA achieved mean room utilization of 0.7800 compared with 0.6085 for Standard GA, an improvement of 0.1715 or 17.15 percentage points (28.18% relative). MCGA also reduced mean cohort-day time gaps from 11.85 to zero. Thus, achieving hard feasibility alone did not ensure efficient room use or compact scheduling [9], [10], [15].

Table 5. Final performance over 20 independent paired runs

Metric	Standard GA	MCGA
Feasibility rate	100%	100%
Mean hard pair conflicts	0.00	0.00
Mean capacity excess*	0.00	0.00
Room utilization	0.6085 ± 0.0130	0.7800 ± 0.0096
Time gaps	11.85 ± 2.56	0.00 ± 0.00
Execution time (s)	150.53 ± 1.49	148.47 ± 1.29

*Capacity excess and room utilization were evaluated for course offerings with validated class-size data (129/141; 91.49%). No capacity excess was observed within this validated subset.

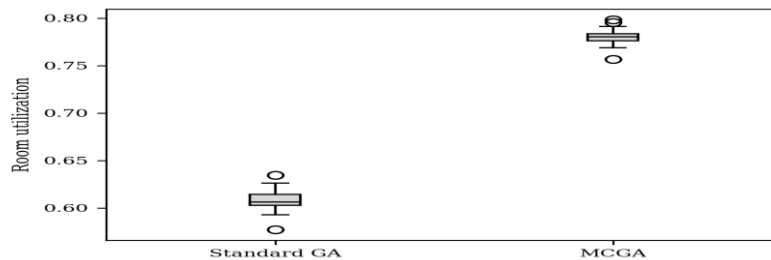


Fig. 2. Room-utilization distribution across the 20 final runs.

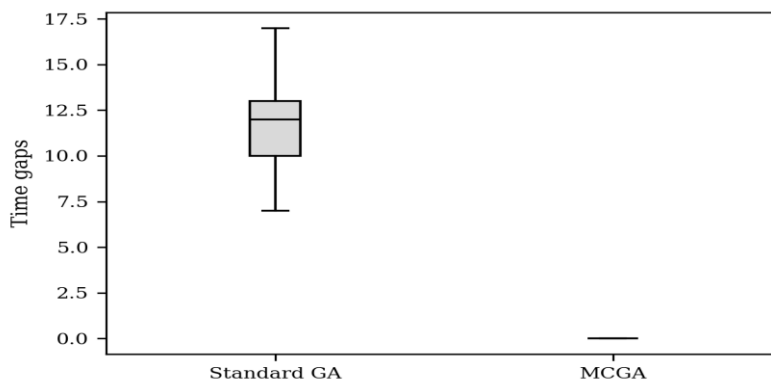


Fig. 3. Time-gap distribution across the 20 final runs.

3.3. Paired Statistical Validation

MCGA improved room utilization and reduced time gaps in all 20 paired runs. The mean room-utilization improvement was 0.1715 (95% CI: 0.1640-0.1790). Wilcoxon signed-rank tests confirmed significant differences in room utilization ($p = 1.91\text{e-}06$) and time-gap reduction ($p = 8.45\text{e-}05$), supporting the robustness of the observed improvements across random seeds [28].

Table 6. Paired effects and statistical tests

Paired metric	Mean difference	95% CI	Wilcoxon p
Room utilization (MCGA - Standard)	0.1715	[0.1640, 0.1790]	1.91e-06
Time-gap reduction	11.85	-	8.45e-05
Runtime (MCGA - Standard)	-2.06 s	-	7.08e-04

3.4. Convergence Behavior

For representative seed 101, both algorithms started with 49 hard conflicts. Standard GA reached zero conflicts at generation 46 but remained at room utilization 0.6053 with 15 time gaps. MCGA reached feasibility at generation 55 with utilization 0.6286 and 11 gaps, reduced gaps to zero by generation 162, and improved final utilization to 0.7985. Thus, while the common mechanisms achieved feasibility, the MCGA weighted fitness continued improving schedule quality within the feasible region [12], [13], [29]. In the proposed design, however, the common repair did not optimize the soft objectives; the continued improvement was driven by the MCGA weighted post-feasibility fitness.

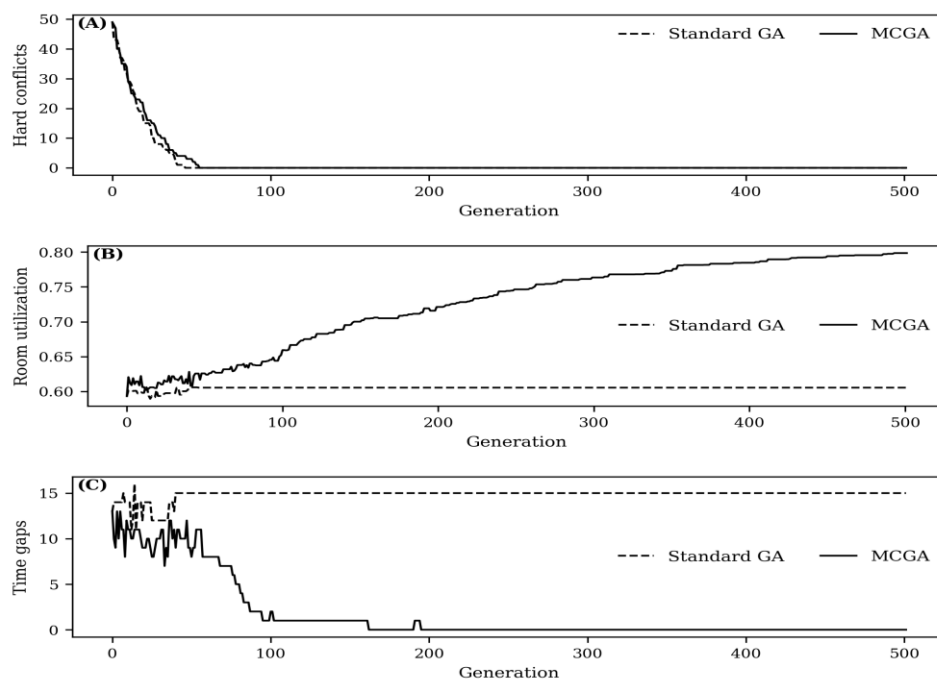


Fig. 4. Representative seed-101 convergence: (A) hard conflicts, (B) room utilization, and (C) time gaps.

3.5. Computational Cost and Practical Interpretation

Mean execution time was 150.53 s for Standard GA and 148.47 s for MCGA, a paired difference of -2.06 s. Although statistically detectable, the difference was only about 1.37% of the Standard GA mean and was considered practically small and environment-dependent. Therefore, MCGA improved room efficiency and time compactness without a material computational-time penalty.

3.6. Answers to the Research Questions

RQ1 was addressed through the day-slot-room chromosome, four hard constraints, capacity-feasible room domain, hierarchical feasibility domination, common hard-only repair, and two soft objectives. RQ2 showed that both algorithms achieved modeled feasibility, while MCGA produced higher room utilization

and zero time gaps. RQ3 showed that mutation rate had the clearest effect among the tested parameter levels, with $MR = 0.02$ producing the best combined schedule quality.

IV. CONCLUSION

This study developed MCGA for university course scheduling by separating hard-feasibility handling from post-feasibility quality optimization. Across 20 paired runs, both Standard GA and MCGA eliminated lecturer, room, and cohort conflicts, with zero capacity excess in the validated subset. MCGA increased mean room utilization from 0.6085 to 0.7800 and reduced mean cohort-day time gaps from 11.85 to zero. The improvements were statistically significant and were achieved without a material computational-time penalty. These results show that a conflict-free timetable is not necessarily efficient and that post-feasibility objectives can substantially improve schedule quality.

The study is limited by validated class sizes for 129 of 141 offerings, empirical rather than official room capacities, cohort-level rather than individual enrollment conflicts, unavailable lecturer-preference data, and evaluation at one institution. Future work should incorporate complete class sizes, official capacities, authentic lecturer preferences, individual enrollment data, and additional institutional datasets.

V. ACKNOWLEDGMENTS

The authors would like to thank STMIK Widya Cipta Dharma for providing academic scheduling data and supporting this research.

REFERENCES

- [1] S. Ceschia, L. Di Gaspero, and A. Schaerf, "Educational timetabling: Problems, benchmarks, and state-of-the-art results," *Eur. J. Oper. Res.*, vol. 308, no. 1, pp. 1–18, 2023, doi: 10.1016/j.ejor.2022.07.011.
- [2] 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.
- [3] A. Bashab et al., "Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues," *Comput. Mater. Contin.*, vol. 74, no. 3, pp. 6461–6484, 2023, doi: 10.32604/cmc.2023.034051.
- [4] S. Abdipoor, R. Yaakob, S. L. Goh, and S. Abdullah, "Meta-heuristic approaches for the University Course Timetabling Problem," *Intell. Syst. with Appl.*, vol. 19, p. 200253, 2023, doi: 10.1016/j.iswa.2023.200253.
- [5] N. M. Arratia-Martinez, P. A. Avila-Torres, and J. C. Trujillo-Reyes, "Solving a University Course Timetabling Problem Based on AACSB Policies," *Mathematics*, vol. 9, no. 19, p. 2500, 2021, doi: 10.3390/math9192500.
- [6] M. Mokhtari, M. V. Sarashk, M. Asadpour, N. Saeidi, and O. Boyer, "Developing a Model for the University Course Timetabling Problem: A Case Study," *Complexity*, vol. 2021, p. 9940866, 2021, doi: 10.1155/2021/9940866.
- [7] E. Rappos, E. Thiémond, S. Robert, and J.-F. Hêche, "A mixed-integer programming approach for solving university course timetabling problems," *J. Sched.*, vol. 25, no. 4, pp. 391–404, 2022, doi: 10.1007/s10951-021-00715-5.
- [8] 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, 2024, doi: 10.1016/j.eswa.2024.123383.
- [9] F. Dunke and S. Nickel, "A matheuristic for customized multi-level multi-criteria university timetabling," *Ann. Oper. Res.*, vol. 328, pp. 1313–1348, 2023, doi: 10.1007/s10479-023-05325-2.
- [10] M. Davison, A. Kheiri, and K. G. Zografos, "Modelling and solving the university course timetabling problem with hybrid teaching considerations," *J. Sched.*, vol. 28, no. 2, pp. 195–215, 2025, doi: 10.1007/s10951-024-00817-w.
- [11] E. Steiner, U. Pfersch, and A. Schaerf, "Curriculum-based university course timetabling considering individual course of studies," *Cent. Eur. J. Oper. Res.*, vol. 33, pp. 277–314, 2025, doi: 10.1007/s10100-024-00923-2.
- [12] 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," *Appl. Intell.*, vol. 51, no. 1, pp. 467–492, 2021, doi: 10.1007/s10489-020-01833-x.
- [13] T. Song, M. Chen, Y. Xu, D. Wang, X. Song, and X. Tang, "Competition-guided multi-neighborhood local

- search algorithm for the university course timetabling problem,” *Appl. Soft Comput.*, vol. 110, p. 107624, 2021, doi: 10.1016/j.asoc.2021.107624.
- [14] M. H. Cruz-Rosales et al., “Metaheuristic with Cooperative Processes for the University Course Timetabling Problem,” *Appl. Sci.*, vol. 12, no. 2, p. 542, 2022, doi: 10.3390/app12020542.
- [15] X. Han and D. Wang, “Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming,” *Algorithms*, vol. 18, no. 3, p. 158, 2025, doi: 10.3390/a18030158.
- [16] L. Yan and H. Deng, “Adaptive genetic algorithm for computer course scheduling system under the background of educational informatization,” *Syst. Soft Comput.*, vol. 7, p. 200324, 2025, doi: 10.1016/j.sasc.2025.200324.
- [17] K. Khaeroni, B. Muqdamien, and A. Hestiningtyas, “Towards Fair and Efficient Timetabling: A Genetic Algorithm Model Integrating Lecturer Day-Off Requests,” *IJID (International J. Informatics Dev.)*, vol. 14, no. 1, pp. 575–586, 2025, doi: 10.14421/ijid.2025.5067.
- [18] R. W. Respati and D. Ramadhani, “Optimizing Lecture Scheduling Using Genetic Algorithm: A Case Study at Universitas Riau,” *Int. J. Electr. Energy Power Syst. Eng.*, vol. 8, no. 2, pp. 220–232, 2025, doi: 10.31258/ijeepse.8.2.220-232.
- [19] T. B. Pongsumarre, W. Wahyuni, and M. Fahmi, “Course Schedule Optimization Using a Java-Based Ant Colony Optimization,” *Media J. Inform.*, vol. 17, no. 2, pp. 328–339, 2025, doi: 10.35194/mji.v17i2.5961.
- [20] R. Salman, S. Suprpto, and Irfandi, “Pengaruh Probabilitas Crossover Terhadap Kinerja Algoritma Genetika Dalam Optimasi Penjadwalan Matakuliah,” *J. Teknoif Tek. Inform. Inst. Teknol.* Padang, vol. 11, no. 2, pp. 69–74, 2023, doi: 10.21063/jtif.2023.V11.2.69-74.
- [21] B. S. G. López, R. A. G. Hernández, and Y. Ledeneva, “Personal Course Timetabling for University Students based on Genetic Algorithm,” *Int. J. Comb. Optim. Probl. Informatics*, vol. 12, no. 3, pp. 32–49, 2021, doi: 10.61467/2007.1558.2021.v12i3.237.
- [22] A. R. Mahlous and H. Mahlous, “Student timetabling genetic algorithm accounting for student preferences,” *PeerJ Comput. Sci.*, vol. 9, p. e1200, 2023, doi: 10.7717/peerj-cs.1200.
- [23] N. M. Arratia-Martinez, C. Maya-Padron, and P. A. Avila-Torres, “University Course Timetabling Problem with Professor Assignment,” *Math. Probl. Eng.*, vol. 2021, p. 6617177, 2021, doi: 10.1155/2021/6617177.
- [24] M. Comlan and C. Allohoumbo, “Constraint satisfaction algorithms: edition of timetables in the license-master-doctorate system,” *Comput. Sci. Inf. Technol.*, vol. 4, no. 3, pp. 217–226, 2023, doi: 10.11591/csit.v4i3.pp217-226.
- [25] D. A. Salem, “University Examination Timetable Scheduling Using Constructive Heuristic Compared to Genetic Algorithm,” *Fayoum Univ. J. Eng.*, vol. 6, no. 1, pp. 39–46, 2023, doi: 10.21608/FUJE.2022.157195.1018.
- [26] I. A. Abduljabbar and S. M. Abdullah, “An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem,” *Baghdad Sci. J.*, vol. 19, no. 2, pp. 399–408, 2022, doi: 10.21123/bsj.2022.19.2.0399.
- [27] T. Müller, H. Rudová, and Z. Müllerová, “Real-world university course timetabling at the International Timetabling Competition 2019,” *J. Sched.*, vol. 28, pp. 247–267, 2025, doi: 10.1007/s10951-023-00801-w.
- [28] R. P. Badoni et al., “An Exploration and Exploitation-Based Metaheuristic Approach for University Course Timetabling Problems,” *Axioms*, vol. 12, no. 8, p. 720, 2023, doi: 10.3390/axioms12080720.
- [29] H. P. Kurniawan, L. Farhatuaini, N. Bahiyah, A. Susanto, M. I. Wilsa, and G. K. Nufus, “Elite-Refined Genetic Algorithm with Hill Climbing Local Search for University Course Scheduling,” *J. Sist. Cerdas*, vol. 8, no. 3, pp. 317–331, 2025, doi: 10.37396/jsc.v8i3.584.
- [30] O. Haitan and O. Nazarov, “Hybrid Approach to Solving of the Automated Timetabling Problem in Higher Educational Institution,” *Syst. Control. Navig. Commun.*, vol. 2, no. 60, pp. 60–69, 2020, doi: 10.26906/SUNZ.2020.2.060.