International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(6s) 2026 pp. 486-497
https://doi.org/10.70917/ijcisim-2026-2950
Article

A Dual-Phase Hybrid Method for Curriculum-Based Academic Scheduling Utilizing Backtracking and Genetic Algorithms

Anand Jawdekar[1], Leeladhar Chourasiya[2], Vikram Kaushik[3], Vicky Gupta[4], Sanjay Pagare[5], Sanjay Patsariya[6], Vivek Gupta[7], Vivek Tiwari[8]

[1]Department of Computer Science and Engineering, Parul University Vadodara, India
Email: anand.jawdekar35064@paruluniversity.ac.in
[2]Department of Computer Science and Engineering, Acropolis Institute of Technology and Research Indore, India;
Email: mhowwala12@gmail.com
[3]Department of Computer Science and Engineering, Parul Institute of Engineering and Technology, Parul University Vadodara, India; Email: dr.vikramkaushik@gmail.com
[4]Department of Computer Science and Engineering, Jaypee Institute of Information Technology (JIIT)
Noida, India; Email: vicky.gupta@mail.jiit.ac.in
[5]Department of Computer Science and Engineering, Parul Institute of Engineering and Technology, Parul University Vadodara, India; Email: sanjay.pagare36802@paruluniversity.ac.in
[6]Department of Computer Science and Engineering, Rustamji Institute of Technology
Gwalior, India; Email: sanjaypatsariya@gmail.com
[7]Department of Computer Science and Engineering, Rustamji Institute of Technology
Gwalior, India; Email: vivekguptachp@gmail.com
[8]Department of Computer Science and Engineering, Parul Institute of Engineering and Technology, Parul University Vadodara, India; Email: vivek642@gmail.com

Abstract: — The generation of academic timetables represents a complex combinatorial optimization challenge that entails the allocation of lectures, practical sessions, instructors, and resources while adhering to various institutional constraints. In programs based on curricula, practical sessions introduce further complications due to laboratory capacity restrictions, necessitating the division of students into batches that must be coordinated within the same academic year. Traditional single-stage methods utilizing Genetic Algorithms often encounter heightened search space and computational complexity when attempting to manage both lecture and practical scheduling concurrently. To mitigate this challenge, this paper introduces a hybrid two-phase timetable generation framework that integrates deterministic backtracking with Genetic Algorithms. In the initial phase, practical sessions are organized using a backtracking algorithm to meet constraints related to laboratory capacity, faculty availability, and batch synchronization. After achieving a viable practical timetable, an occupancy matrix is created, and the remaining lecture sessions are optimized through a Genetic Algorithm that takes into account both hard and soft constraints. This proposed method simplifies the search process by distinguishing between highly constrained and less constrained scheduling elements. Experimental assessments conducted on a curriculum-based dataset comprising three academic years, 12 faculty members, 16 lecture subjects, and 12 practical subjects reveal that the proposed framework produces conflict-free timetables with efficient execution times. For a problem size of 85 events, a complete timetable was generated in under 4 seconds. The findings suggest that the proposed methodology offers a scalable and computationally efficient solution for medium-sized academic departments and serves as a practical alternative to traditional single-stage timetable generation techniques.

Keywords: — Timetable scheduling, curriculum-based scheduling, Genetic Algorithm (GA), Backtracking algorithm, Laboratory scheduling

1. Introduction

The generation of academic timetables is a crucial scheduling activity within educational institutions, entailing the allocation of courses, instructors, classrooms, laboratories, and time slots while adhering to various institutional and academic constraints. In programs based on curricula, timetables for several academic years are created concurrently, with resources such as faculty, classrooms, and laboratories being utilized across different years. Given that the University Course Time tabling Problem (UCTP) is classified as NP-hard, the task of producing feasible and high-quality timetables becomes increasingly challenging as the institution's size and the number of constraints grow.

The conventional method of preparing timetables manually is not only labor-intensive and time-consuming but also highly susceptible to conflicts and inefficiencies. Over time, a range of optimization techniques has been introduced to automate this process, including Integer Programming, Constraint Programming, Genetic Algorithms (GA), Simulated Annealing, Tabu Search, and various other meta-heuristic strategies. Among these techniques, Genetic Algorithms have shown notable effectiveness due to their ability to navigate extensive search spaces and yield near-optimal solutions. Nevertheless, employing a single-stage Genetic Algorithm to concurrently schedule lectures and laboratory sessions often leads to heightened computational complexity and slower convergence, particularly when practical sessions necessitate synchronization constraints and limited laboratory resources.

A major challenge emerges in curriculum-based undergraduate programs where students are organized into batches for practical sessions due to restrictions on laboratory capacity. While different batches participate in separate laboratory sessions, all batches from the same academic year must engage in practical sessions at the same time to facilitate their attendance at common lectures. This requirement for synchronization introduces additional hard constraints, rendering the scheduling of practical sessions significantly more complex than that of lectures.

2. Related work

Timetable generation in universities is modeled as the University Course Timetabling Problem (UCTP), which involves assigning courses to time slots, rooms, and instructors while avoiding conflicts and satisfying institutional constraints. It is mostly divided into Curriculum-Based Course Timetabling (CB-CTT) and Post-Enrollment Course Timetabling (PE-CTT). This study is on the curriculum-based setting, where each academic year follows a fixed set of subjects.

The problem is considered NP-hard because multiple constraints must be handled simultaneously, and its complexity increases with institutional size [1]. Early approaches relied on manual scheduling and simple heuristics, but later research applied mathematical models such as Integer Programming and Constraint Programming [2]. As problem size grew, metaheuristic methods like Genetic Algorithms, Tabu Search, and Simulated Annealing became popular for producing near-optimal solutions efficiently [3]. Genetic Algorithms, in particular, showed strong results in benchmark studies such as the International Time tabling Competition (ITC-2007) [4].

One persistent challenge in departmental scheduling is the allocation of practical sessions. Unlike lecture sessions, laboratory sessions require labs with limited capacity. In multi-year undergraduate programs, accommodating all student batches within available laboratory resources increases the scheduling complexity. Many systems handle lectures and practical sessions simultaneously, which expands the search space and can reduce overall efficiency.

To manage this complexity, hybrid strategies that combine deterministic and evolutionary methods have been proposed [5]. Deterministic approaches, such as backtracking, are well suited for tightly constrained tasks like laboratory allocation, whereas evolutionary methods like Genetic Algorithms are effective for optimizing lecture scheduling and balancing workloads. The present work adopts such a hybrid framework to produce practical and balanced timetables for multi-year, curriculum-based academic programs.

Recent studies have shown a growing interest in sophisticated optimization methods for addressing the University Course Timetabling Problem (UCTP). Hyper-heuristic strategies have gained prominence due to their capacity to selectively and adaptively choose low-level heuristics, thereby producing high-quality solutions across various problem instances [12]. Memetic algorithms, which integrate evolutionary search with local enhancement techniques, have exhibited enhanced convergence properties and a decrease in constraint violations [10]. Additionally, swarm intelligence methods such as Particle Swarm Optimization (PSO) [14] and Ant Colony Optimization (ACO) [13] have been effectively utilized to navigate extensive search spaces with efficiency. Moreover, recent research has

concentrated on hybrid meta-heuristic frameworks and decomposition-based strategies to tackle large-scale timetabling challenges [7], [8]. In more recent developments, reinforcement learning methods have surfaced as a promising avenue for combinatorial optimization issues, providing adaptive decision-making capabilities and potential relevance to dynamic timetabling situations [11].

3. Proposed methodology

A. System Overview

The university timetabling problem considered in this work follows a curriculum-based structure where each academic year operates under a fixed semester schedule. Each year consists of a set of lecture subjects, practical subjects, laboratory batches, and a predefined weekly timetable grid.

A specific constraint in the institution is that all batches belonging to the same academic year must conduct their practical sessions at the same time, as the students in same year can attend lectures together. Because of this synchronization requirement, the scheduling process becomes more restrictive than in standard curriculum-based timetabling problems.

The scheduling model uses several institutional and academic parameters, including faculty availability, classroom and laboratory resources, student enrollment strength, batch size limits, and the required number of teaching hours for each subject per week. The institutional working time grid is used for calculating available time slots. These parameters together determine the feasible space for timetable generation.

B. Constraint Modelling

To generate a valid timetable, the system considers two types of constraints: hard constraints and soft constraints.

Hard constraints are mandatory and must always be satisfied.

Global Hard Constraints:

Applied on both lecture and practical scheduling.

- A teacher cannot be assigned to more than one session in the same time slot.
- A classroom or laboratory cannot host more than one session at the same time.
- A student cannot attend more than one session simultaneously.
- Room or laboratory capacity must be greater than or equal to the number of assigned students.
- Once a subject or practical is assigned to a teacher, the same teacher must handle it for the entire timetable.

Practical-Specific Hard Constraints:

The following constraints apply only to practical sessions.

- If a practical slot is assigned to an academic year, all batches of that year must have practical sessions.
- Once a laboratory is allocated to a batch, the same laboratory is used for all practical sessions of that batch.
- The required weekly practical hours for each subject must be satisfied exactly.
- A maximum of two practical sessions can be assigned to a batch in a single day.

Soft Constraints:

Applied only on lectures. Soft constraints are not mandatory but help improve the quality of the timetable. Violations of these constraints are penalized during optimization.

- Minimising free slots between lectures
- Maintaining balanced teaching loads for faculty members
- Distributing lectures evenly across the week

C. Lecture Scheduling

The fitness function is used to evaluate each individual

Fitness Function

$$\text{Fitness}=W_h(HC)+W_s(SC) \quad (1)$$

where

- HC = number of hard constraint violations
- SC = number of soft constraint violations
- $W_h \gg W_s$

D. Two-Phase Scheduling Strategy

The scheduling process is divided into two stages. First, practical sessions are assigned using a backtracking algorithm. Then, the remaining lecture sessions are scheduled using a Genetic Algorithm (GA).

Practical sessions are handled first because they require strict coordination of batches, laboratories, and instructors, which significantly limits feasible scheduling options. A backtracking approach ensures that these sessions are assigned without conflicts, reducing the search space for phase two.

Once a feasible practical timetable is found, those time slots are fixed and recorded. The Genetic Algorithm then schedules lectures in the remaining slots around the practical sessions. The overall workflow of the proposed hybrid scheduling strategy is illustrated in Figure 1.

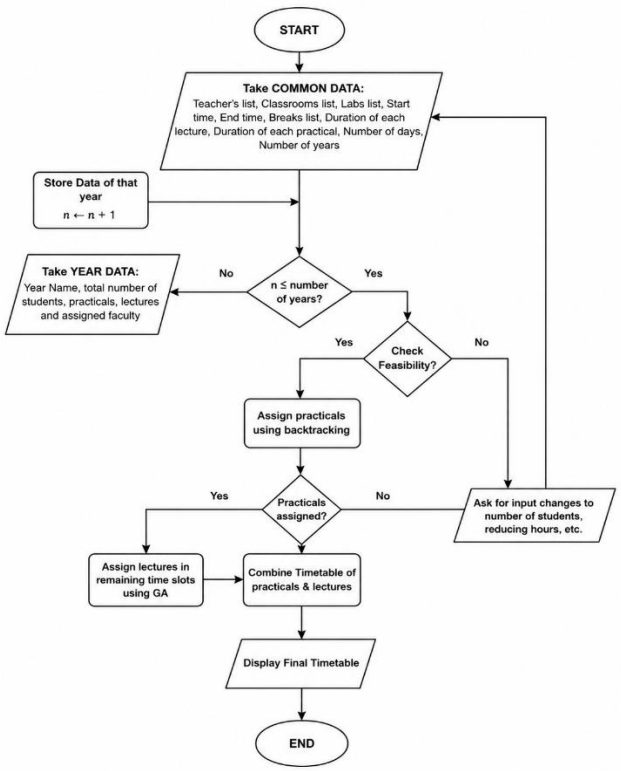

Fig. 1. Flowchart of the Proposed Approach

Practical Scheduling:

In this phase, only practical sessions are scheduled. The synchronized scheduling of batches makes the practical scheduling more difficult. For example, let's take a single academic year/class with 60 students and average lab capacity of 20. Due to the limited lab capacity and for effective management, the 60 students are divided into 3 batches (B1, B2, B3) each containing 20 students for only practical sessions. This means that all batches must have practical sessions at the same time so that they can attend the lectures together. But each batch can have a different practical session in separate labs with different instructors.

A Single practical time slot of a year for this example is shown in Table I.

Table 1. Example of one slot of practical

Batch	Practical	Lab	Teacher
B1	AIML Lab	L1	F1
B2	IP Lab	L2	F2
B3	CN Lab	L3	F3

Multiple such slots are assigned in the practical timetable until all batches complete each practical's required number of hours given by the user. The program keeps track of the number of sessions assigned to the batch and continues allocating practical slots until all the constraints are met.

We used a simple backtracking algorithm to create the practicals timetable. Then the time slots which are occupied by practical's are removed from the available timeslots and a matrix is created to track teachers, labs occupancy which is used by the GA.

In some cases, it is not possible to generate a valid practical timetable due to insufficient resources. Therefore, first a basic feasibility check is done to determine if the resources given by the user are feasible to generate the practical timetable as well as the lecture one.

Lecture Scheduling:

After fixing the practical timetable, lectures are scheduled using Genetic Algorithm in remaining free time slots. In this phase, each chromosome represents a possible lecture slot containing information about the subject, instructor, classroom, and time slot. The initial population is generated randomly while respecting the constraints such as number of hours of each lecture and faculty eligibility set by the user. It also avoids assigning lectures with teachers occupied in practicals of other academic years, which reduces the convergence time of the Genetic Algorithm.

An individual is a complete timetable of all academic years. The fitness function is used to evaluate each individual, based on the hard constraints and additional soft constraints. Hard constraints are given higher penalties as they should be always satisfied to create a valid usable timetable and soft constraints are given lower penalty. Soft constraints improve the overall quality of the timetable. We assigned higher weight to the soft constraint of minimizing free slots between non-free slots among the other soft constraints.

The GA uses standard evolutionary operators including tournament selection, one point crossover which chooses a random day as the point of crossover between the two parents. Elitism is applied so that the best solutions from one generation are carried forward and not lost.

After the program reaches the set number of generations or a set fitness score, it is stopped and the final timetable is shown along with a bar chart to visualize teachers hours assigned over the entire timetable, which helps user improve the practical-faculty pairing.

Algorithm 1 Practical Scheduling using Backtracking

Input : Practical subjects, labs, teachers

Output : Practical timetable

For each practical

Check constraints

Assign slot

If conflict occurs

Backtrack

Return timetable

Algorithm 2 Lecture Scheduling using Genetic Algorithm

Initialize population

Evaluate fitness

Repeat

Selection

Crossover

Mutation

Elitism

Until stopping criterion

Return best timetable

E. Justification of Two phases

Traditional single stage Genetic Algorithm approaches for timetable generation are effective in finding feasible timetables, but the time taken by them is also more. The practical scheduling for batches introduce a lot of constraints, which increases the time taken by GA to find a solution. To make the timetable generation faster, we used the two phases.

The approach separates the problem by first handling the more difficult and constrained part of practicals using a deterministic method of backtracking. By fixing the practicals before, the overall search space is significantly reduced, allowing the Genetic Algorithm to operate more effectively for lecture scheduling. This increases the efficiency of the search process.

4. Experimental Setup

A. Dataset description

The evaluation dataset is of a curriculum-based academic program consisting of three different academic years. Lecture subjects requiring on average 4 sessions weekly and each lecture session is of 1 hour, while practical subjects are conducted once per week occupying 2 hours but divided into multiple batches due to laboratory capacity limitation. Table II and Table III show the important details about the dataset and the parameters set for Genetic Algorithm respectively.

The problem size is the sum of total lecture sessions and practical batch sessions of all academic years. Execution time is measured from start of the practical scheduling phase till the lecture scheduling phase and generation of the complete timetable obtained by combining the two.

Table 2. Dataset parameters

Parameter	**Value**
Number of Academic Years to generate timetables of	3
Total Number of Teachers	12
Total Labs	4
Total Classrooms	3
Total Lecture Subjects	16
Total Practical Subjects	12
Practical batches/year	3
Total Weekly timetable slots for each academic year	30

Table 3. Genetic algorithm parameters

Parameter	**Value**
Population size	50

Generations	200
Crossover Rate	0.8
Mutation Rate	0.2
Selection Method	Tournament selection
Crossover Method	One point crossover

5. Results and Discussion

A. Execution Time Analysis

Table IV presents the execution time required to generate timetables of different sizes derived from all three academic years.

Table 4. execution time analysis table

Academic Year	**Practical Events**	**Lecture Events**	**Problem Size**	**Practical Scheduling Time (s)**	**Lecture Scheduling Time (s)**	**Total Time (s)**
Only Year 1	15	15	30	0.12	1.45	1.57
Only Year 2	9	21	30	0.25	1.68	1.93
Only Year 3	12	13	25	0.08	1.41	1.42
All Years	36	49	85	0.28	3.57	3.85

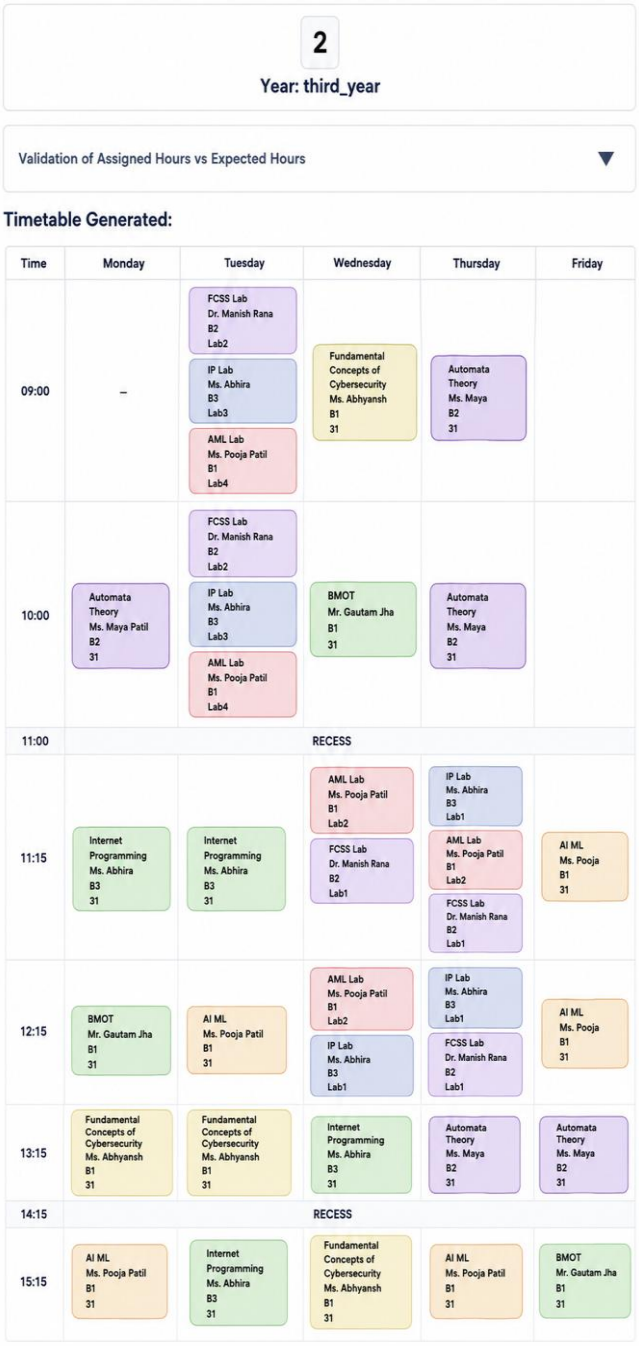

Fig. 2. Department dashboard showing input parameters for timetable generation

The results show that timetable generation remains computationally efficient even as the problem size increases. Practical scheduling is faster as we are scheduling them first without any lectures and backtracking algorithm is able to find a solution easily. The lecture scheduling phase is slower as it uses GA to arrange lecture sessions in between the practical time slots using multiple generations.

When the timetable for all 3 years was generated simultaneously, the problem size increased to 85 and even then a complete timetable that satisfied the constraints was produced under 4 seconds, indicating that the proposed methodology scales reasonably for medium sized departments.

Comparing to a standard Genetic Algorithm that schedules both lectures and practical together is expected to struggle with strict synchronization constraints, leading to larger infeasible areas in the search space. Whereas the proposed approach reduces the complexity by separating the practical scheduling, resulting in faster convergence.

B. Performance Metrics

Constraint Satisfaction Rate (CSR):

$$\text{CSR}= \frac{\text{Total Constraints Satisfied}}{\text{Constraints}} \times 100 \quad (2)$$

Resource Utilization:

$$\text{Teacher Utilization} \quad \text{TU} = \frac{Assigned\ Hour}{Available\ Hour} \times 100 \quad (3)$$

$$\text{Laboratory Utilization} \quad \text{LU} = \frac{Used\ Lab\ Slots}{Total\ Lab\ Slots} \times 100 \quad (4)$$

Table V: COMPARISON TABLE

Metric	**Value**
Constraint Satisfaction	100%
Teacher Utilization	82%
Lab Utilization	76%
Classroom Utilization	69%

C. Comparison with Conventional Genetic Algorithm

Table VI: COMPARISON TABLE

Method	**Problem Size**	**Time (s)**	**Constraint Violations**
Standard GA	85	7.31	3
Proposed Two-Phase	85	3.85	0

In order to assess the efficacy of the suggested methodology, a comparison was conducted with a traditional single-stage Genetic Algorithm, where practical and lecture sessions were scheduled concurrently.

The findings demonstrate that the proposed two-phase strategy resulted in quicker convergence and minimized the search space by pre-scheduling highly constrained practical sessions. For the entire dataset spanning three academic years, the proposed method produced conflict-free timetables in 3.85 seconds, while the conventional Genetic Algorithm necessitated a longer computational duration.

D. System Output

The system provides a user friendly interface for managing and generating timetables. Fig. 2 shows the department page.

The project was implemented using Python along with DEAP library for implementing Genetic Algorithm. The frontend was created using HTML, CSS, Javascript. The backend was created using Django framework and SQLite was used for database which is configured in Django.

Fig. 3. Generated timetable of one single year

The User interface is divided into four pages. First page being the dashboard where we show all the department's resources such as timeslots, recess time slots, teachers, classrooms, laboratories, and academic years added. The second page is for managing years which allows users to add new years or change data from existing years added by them. It also displays a pie chart that visualizes the distribution of time across categories of recess hours, free hours, practical hours, lecture hours. The third page allows users to actually generate the timetable for that department and also displays timetables saved by the users. The final fourth page is for displaying the generated or saved timetable which also shows a bar chart of assigned hours vs teachers that visualizes teachers workload across all academic years.

When timetable for all 3 years was generated simultaneously, the problem size increased to 85 and despite that complete timetable that satisfied the constraints was produced under 4 seconds, indicating that the proposed methodology scales reasonably for medium sized departments. The generated timetable scheduled the practicals and lectures effectively without any conflicts. Fig. 3 shows the timetable of one single academic year.

E. Convergence Behaviour of Genetic Algorithm

Fitness value decreases over generations.

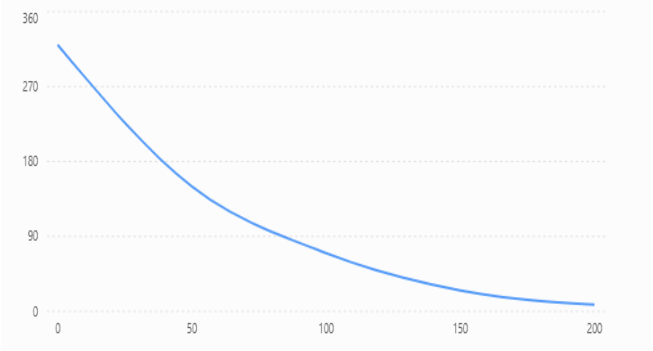

Fig. 4. Convergence behavior of the Genetic Algorithm

Figure 4 shows the convergence behavior of the Genetic Algorithm. The fitness value decreases steadily over generations, indicating that the algorithm effectively explores the search space and converges towards high-quality solutions.

F. Scalability Analysis

Table VII: COMPARISON TABLE

Problem Size	Total Time (s)
30	1.57
60	2.65
85	3.85
120	5.92
150	8.31

As the problem size increases, execution time increases approximately linearly, demonstrating that the proposed methodology scales effectively for medium-sized academic departments.

G. Limitations

Despite its computational efficiency, the proposed approach has certain limitations. Fixing practical sessions before lecture scheduling reduces the flexibility available to the Genetic Algorithm. Furthermore, the methodology has been evaluated only on medium-sized datasets and its scalability for larger institutions has not yet been investigated. Dynamic timetable modifications and real-time rescheduling are also outside the scope of this study.

But the lecture scheduling had a large number of possible arrangements and was less constrained than the practicals. This allowed the Genetic Algorithm to effectively explore the remaining search space and generate feasible timetables despite having reduced flexibility.

This trade-off reflects the decision to prioritize the feasibility of high constrained components over the flexibility of less constrained ones.

6. Conclusion and Future Work

In this study we focused on the problem of curriculum based timetable generation, where multiple academic years share common resources such as teachers, classrooms, and laboratories. In this the practical sessions were more difficult to manage due to limited laboratory capacity which leads to the need to divide each academic year into batches only for practicals. This differentiates the practicals from lectures because of the need to coordinate multiple batches at the same time.

To address these challenges, a two-phase approach is used. In the first phase, practical sessions are scheduled using a backtracking method to ensure that all practical specific constraints are satisfied. Once a practical timetable is found, it is fixed and then lectures are scheduled in the remaining time slots using a Genetic Algorithm.

Results show that this approach is able to generate feasible timetables in less time while satisfying all required constraints. By separating the hard constrained part with less constrained part leads to reduction of overall complexity.

Future work will focus on comparing the proposed methodology with conventional Genetic Algorithms and other metaheuristic techniques. Hyper-heuristic and memetic approaches may also be investigated to improve solution quality. In addition, dynamic rescheduling capabilities and multi-objective optimization strategies will be explored to enhance applicability in large-scale academic institutions.

References

1. A. Schaerf, "A survey of automated timetabling," Artificial Intelligence Review, vol. 13, no. 2, pp. 87–127, 1999.
2. E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," European Journal of Operational Research, vol. 140, no. 2, pp. 266–280, 2002.
3. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.
4. E. K. Burke et al., "The second international timetabling competition (ITC-2007): Curriculum-based course timetabling," Annals of Operations Research, vol. 194, no. 1, pp. 7–30, 2012.
5. S. Abdullah, E. K. Burke, and B. McCollum, "A hybrid evolutionary approach to the university course timetabling problem," IEEE Transactions on Evolutionary Computation, vol. 16, no. 1, pp. 86–98, 2012.
6. S. Ahmadi et al., "Meta-heuristic approaches for the university course timetabling problem: A survey," Computers & Operations Research, 2023.
7. 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, 2023, doi: 10.1016/j.iswa.2023.200253.
8. J. Almeida, J. R. Figueira, A. P. Francisco, and D. Santos, "A Hybrid Meta-Heuristic for the Generation of Feasible Large-Scale Course Timetables Using Instance Decomposition," arXiv preprint arXiv:2310.20334, 2023.
9. A. Akkan and B. Gülcü, "A bi-criteria hybrid genetic algorithm with robustness objective for the course timetabling problem," Computers & Operations Research, vol. 90, pp. 22–32, 2018, doi: 10.1016/j.cor.2017.08.014.
10. E. K. Burke, J. P. Newall, and R. F. Weare, "A Memetic Algorithm for University Exam Timetabling," Lecture Notes in Computer Science, vol. 1153, pp. 241–250, Springer, 1996.
11. X. Chen, Y. Wang, and Z. Li, "Deep Reinforcement Learning for Combinatorial Optimization Problems: A Survey," IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 2, pp. 1540–1558, 2024, doi: 10.1109/TNNLS.2022.3148144.
12. E. K. Burke, G. Kendall, and E. Soubeiga, "A Hyper-Heuristic Approach to Automated Timetabling,"Lecture Notes in Computer Science, vol. 2079, pp. 181–194, Springer, 2001.
13. M. Dorigo and T. Stützle, "Ant Colony Optimization: Overview and Recent Advances,"Handbook of Metaheuristics, Springer, pp. 311–351, 2019.
14. J. Kennedy and R. Eberhart,"Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948, 1995, doi: 10.1109/ICNN.1995.488968.